Learning with Opponent-Learning Awareness

Jakob N. Foerster[2,†] jakob.foerster@cs.ox.ac.uk
Richard Y. Chen[1,†] richardchen@openai.com
Maruan Al-Shedivat[4] alshedivat@cs.cmu.edu
Shimon Whiteson[2] shimon.whiteson@cs.ox.ac.uk
Pieter Abbeel[3] pieter@openai.com
Igor Mordatch[1] mordatch@openai.com

[1]OpenAI [2]University of Oxford [3]UC Berkeley [4]CMU

Abstract

Multi-agent settings are quickly gathering importance in machine learning. Beyond a plethora of recent work on deep multi-agent reinforcement learning, hierarchical reinforcement learning, generative adversarial networks and decentralized optimization can all be seen as instances of this setting. However, the presence of multiple learning agents in these settings renders the training problem non-stationary and often leads to unstable training or undesired final results. We present *Learning with Opponent-Learning Awareness* (LOLA), a method that reasons about the anticipated learning of the other agents. The LOLA learning rule includes an additional term that accounts for the impact of the agent's policy on the anticipated parameter update of the other agents. We show that the LOLA update rule can be efficiently calculated using an extension of the likelihood ratio policy gradient update, making the method suitable for model-free reinforcement learning. This method thus scales to large parameter and input spaces and nonlinear function approximators. Preliminary results show that the encounter of two LOLA agents leads to the emergence of tit-for-tat and therefore cooperation in the infinitely iterated prisoners' dilemma, while independent learning does not. In this domain, LOLA also receives higher payouts compared to a naive learner, and is robust against exploitation by higher order gradient-based methods. Applied to infinitely repeated matching pennies, LOLA agents converge to the Nash equilibrium. In a round robin tournament we show that LOLA agents can successfully shape the learning of a range of multi-agent learning algorithms from literature, resulting in the highest average returns on the IPD. We also apply LOLA to a grid world task with an embedded social dilemma using deep recurrent policies. Again, by considering the learning of the other agent, LOLA agents learn to cooperate out of selfish interests.

1 Introduction

Due to the advent of deep RL methods that allow the study of many agents in rich environments, multi-agent reinforcement learning has flourished in recent years. However, most of this work considers fully cooperative settings (Omidshafiei et al., 2017; Foerster et al., 2018, 2017) and emergent communication in particular (Das et al., 2017; Mordatch and Abbeel, 2017; Lazaridou, Peysakhovich, and Baroni, 2016; Foerster et al., 2016; Sukhbaatar, Fergus, and others, 2016). Considering future applications of multi-agent RL, such as self-driving cars, it is obvious that many of these will be only partially cooperative and contain elements of competition and selfish incentives.

The human ability to maintain cooperation in a variety of complex social settings has been vital for the success of human societies. Emergent reciprocity has been observed even in strongly adversarial settings such as wars (Axelrod, 2006), making it a quintessential and robust feature of human life.

In the future, artificial learning agents are likely to take an active part in human society, interacting both with other learning agents and humans in complex partially competitive settings. Failing to develop learning algorithms that lead to emergent reciprocity in these artificial agents would lead to disastrous outcomes.

How reciprocity can emerge among a group of learning, self-interested, reward maximizing RL agents is thus a question both of theoretical interest and of practical importance. Game theory has a long history of studying the learning outcomes in games that contain cooperative and competitive elements. In particular, the tension between cooperation and defection is commonly studied in the iterated prisoners' dilemma. In this game, selfish interests can lead to an outcome that is overall worse for all participants, while cooperation maximizes social welfare, one measure of which is the sum of rewards for all agents.

Interestingly, in the simple setting of an infinitely repeated prisoners' dilemma with discounting, randomly initialized RL agents pursuing independent gradient descent on the exact value function learn to defect with high probability. This shows that current state-of-the-art learning methods in deep multi-agent RL can lead to agents that fail to cooperate reliably even in simple social settings with explicit actions to cooperate and defect. One well-known shortcoming is that they fail to consider the learning process of the other agents and simply treat the other agent as a *static* part of the environment.

As a step towards reasoning over the learning behaviour of other agents in social settings, we propose *Learning with Opponent-Learning Awareness* (LOLA). The LOLA learning rule includes an additional term that accounts for the impact of one agent's parameter update on the learning step of the other agents. For convenience we use the word 'oppo-

[†]Equal Contribution

nent' to describe the other agent, even though the method is not limited to zero-sum games and can be applied in the general-sum setting. We show that this additional term, when applied by both agents, leads to emergent reciprocity and cooperation in the iterated prisoners' dilemma (IPD). Experimentally we also show that in IPD, each agent is incentivized to switch from naive learning to LOLA, while there are no additional gains in attempting to exploit LOLA with higher order gradient terms. This suggests that within the space of local, gradient-based learning rules both agents using LOLA is a stable equilibrium. This is further supported by the good performance of the LOLA agent in a round-robin tournament, where it successfully manages to shape the learning of a number of multi-agent learning algorithms from literature. This leads to the overall highest average return on the IPD, and good performance on IMP.

We also present a version of LOLA adopted to the deep RL setting using likelihood ratio policy gradients, making LOLA scalable to settings with high dimensional input and parameter spaces.

We evaluate the policy gradient version of LOLA on the iterated prisoners dilemma (IPD) and iterated matching pennies (IMP), a simplified version of rock-paper-scissors. We show that LOLA leads to cooperation with high social welfare, while independent policy gradients, a standard multi-agent reinforcement learning approach, does not. The policy gradient finding is consistent with prior work, e.g., Sandholm and Crites (1996). We also extend LOLA to settings where the opponent policy is unknown and needs to be inferred from state-action trajectories of the opponent's behaviour.

Finally, we apply LOLA with and without opponent modelling to a grid-world task with an embedded underlying social dilemma. This task has temporally extended actions and therefore requires high dimensional recurrent policies for agents to learn to reciprocate. Again, cooperation emerges in this task when using LOLA, even when the opponent's policy is unknown and needs to be estimated.

2 Related Work

The study of general-sum games has a long history in game theory and evolution. Thousands of papers have been written on the iterated prisoners' dilemma (IPD) alone, including the seminal work on the topic by Axelrod (2006). This work popularized tit-for-tat (TFT), a strategy in which an agent cooperates on the first move and then copies the opponent's most recent move, as a robust and simple strategy in the IPD.

Learning and stability in sequential, general sum games has also been studied in the multi-agent RL community. Seminal work includes the family of WoLF algorithms (Bowling and Veloso, 2002), which achieve convergence by using different learning rates depending on whether an agent is winning or losing, Joint-action-learners, (Claus and Boutilier, 1998; Banerjee and Sen, 2007) and friend-or-foe learning (Littman, 2001). Wunder, Littman, and Babes (2010), Zinkevich, Greenwald, and Littman (2006) and Sandholm and Crites (1996) explicitly study the convergence dynamics and equilibria of learning in iterated games.

Brafman and Tennenholtz (2003) introduce the solution concept of an 'efficient learning equilibrium' (ELE), in which neither side is encouraged to deviate from the learning rule. The algorithm they propose applies to settings where all Nash equilibria can be computed and enumerated. At the intersection of RL and evolutionary methods, Tuyls et al. (2003) uses replicator dynamics for understanding the convergence points of multi-agent systems.

While it is beyond the scope of this work to list all relevant methods in multi-agent RL, we refer the reader to an excellent review on the subject (Busoniu, Babuska, and De Schutter, 2008). However, we note that multi-agent RL historically focuses on settings without function approximation and often considers methods and feature representations that are highly tuned towards specific problem settings, such as the iterated prisoners dilemma. Recent evaluations by Zawadzki, Lipson, and Leyton-Brown (2014) suggest that independent learning is still one of the most robust and versatile methods.

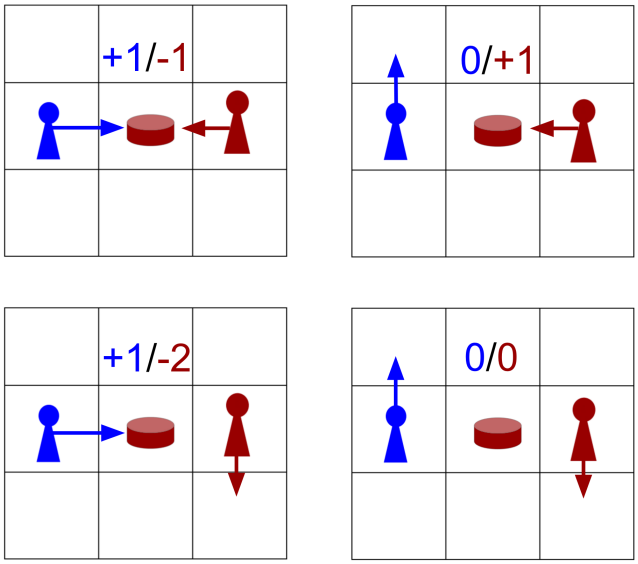


Figure 1: In the coin game, two agents 'red' and 'blue', get 1 point for picking up each coin. However, the 'red agent' loses 2 points when the 'blue agent' picks up a red coin and vice versa. Effectively this is a world with an embedded social dilemma where the action to cooperate and defect are temporally extended.

In contrast, potentially due to the issues mentioned above, most work in deep multi-agent RL focuses on fully cooperative settings (Omidshafiei et al., 2017; Foerster et al., 2018, 2017) and emergent communication in particular (Das et al., 2017; Mordatch and Abbeel, 2017; Lazaridou, Peysakhovich, and Baroni, 2016; Foerster et al., 2016; Sukhbaatar, Fergus, and others, 2016). As an exception, Leibo et al. (2017) consider the outcomes of independent learning in general sum settings. Lowe et al. (2017) propose a centralized actor-critic architecture for efficient training in these mixed environments. However, none of these

methods explicitly reasons about the learning behaviour of other agents. Lanctot et al. (2017) generalize the ideas of game-theoretic best response style algorithms, such as NFSP (Heinrich and Silver, 2016), to produce more general policies. In contrast to LOLA, these best-response-algorithms assume a given set of opponent policies, rather than attempting to shape the learning of the other agents.

The problem setting of Lerer and Peysakhovich (2017) is closest to our setting. They directly generalize tit-for-tat to complex environments using deep RL. The authors explicitly train a fully cooperative and a defecting policy for both agents and then construct a tit-for-tat policy that switches between these two in order to encourage the opponent to cooperate. Similar in spirit to this work, Munoz de Cote and Littman (2008) propose a Nash equilibrium algorithm for repeated stochastic games that explicitly attempts to find the egalitarian point by switching between competitive and zero-sum strategies. A similar point underlies M-Qubed, Crandall and Goodrich (2011), which balances best-response, cautious, and optimistic learning biases.

Reciprocity and cooperation are not emergent properties of the learning rules in these algorithms but rather directly coded into the algorithm. By contrast, LOLA makes no assumptions about cooperation and simply assumes that each agent is maximizing its own return.

Our work also relates to opponent modeling, such as fictitious play (Brown, 1951) and action-sequence prediction (Mealing and Shapiro, 2015). Mealing and Shapiro (2013) also propose a method that finds a policy based on predicting the opponent's future action. Furthermore, Hernandez-Leal and Kaisers (2017) directly model the distribution over opponents. While these methods model the opponent strategy, they do not address the learning dynamics of the opponent.

By contrast, Zhang and Lesser (2010) carry out policy prediction under one-step learning dynamics. However, the opponents' policy updates are assumed to be fixed and only used to learn a best response to the anticipated updated parameters. By contrast, LOLA directly models the policy updates of all opponents such that each agent actively drives its opponents' policy updates to maximize its own reward. Differentiating through the opponent's learning step, which is unique to LOLA, is crucial for the emergence of tit-for-tat and reciprocity. Hernandez-Leal et al. (2017) offer an up-to-date survey of methods addressing the non-stationarity in multi-agent learning.

With LOLA, each agent differentiates its estimated return through the opponents' policy update. Similar ideas were proposed by Metz et al. (2016), whose training method for generative adversarial networks differentiates through multiple update steps of the opponent. Their method relies on an end-to-end differentiable loss function, and thus does not work in the general RL setting. However, the overall results are similar: anticipating the opponent's update stabilises the training outcome.

Outside of purely computational studies the emergence of cooperation and defection in RL settings has also been studied and compared to human data Kleiman-Weiner et al. (2016).

3 Background

Our work assumes a multi-agent task that is commonly described as a stochastic game G, specified by a tuple $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$. Here n agents, $a \in A \equiv \{1, ..., n\}$, choose actions, $u^a \in U$, and $s \in S$ is the state of the environment. The joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$ leads to a state transition based on the transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. The reward functions $r^a(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ specify the reward for each agent, lastly $\gamma \in [0, 1)$ is the discount factor.

We further define the discounted future return from time t onward as $R_t^a = \sum_{l=0}^{\infty} \gamma^l r_{t+l}^a$ for each agent, a. As a naive learner, each agent maximizes its total discounted return in expectation separately. This can be done with policy gradient methods (Sutton et al., 1999) such as REINFORCE (Williams, 1992). Policy gradient methods update an agent's policy, parameterized by $\boldsymbol{\theta}^a$, by performing gradient ascent on an estimate of the expected discounted total reward $\mathbb{E}\left[R_0^a\right]$.

By convention, bold lowercase letters denote column vectors.

4 Methods

In this section, we review the naive learner's strategy and introduce the LOLA learning rule. We first derive the update rules when agents have access to exact gradients and Hessians of their expected discounted future return in Sections 4.1 and 4.2. In Section 4.3, we derive the learning rules based purely based on policy gradients, thus removing access to exact gradients and Hessians. This renders LOLA suitable for deep RL. However, we still assume agents have access to opponent's policy parameters in policy gradient-based LOLA. Next, in Section 4.4, we incorporate opponent modeling into the LOLA learning rule, such that each LOLA agent only infers the opponent's policy parameter from experiences. Finally, we discuss higher order LOLA learning in Section 4.5.

For simplicity, we assume the number of agents is $n = 2$ and display the update rules for agent 1 only. The same derivation holds for arbitrary numbers of agents.

4.1 Naive Learner

Suppose each agent's policy π^a is parameterized by $\boldsymbol{\theta}^a$ and $V^a(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)$ is the expected total discounted return for agent a as a function of both agents' policy parameters $(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)$. A naive learner iteratively optimizes for its own expected total discounted return separately, such that at the ith iteration, $\boldsymbol{\theta}_i^a$ is updated to $\boldsymbol{\theta}_{i+1}^a$ according to

$$\boldsymbol{\theta}_{i+1}^1 = \text{argmax}_{\boldsymbol{\theta}^1} V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}_i^2)$$
$$\boldsymbol{\theta}_{i+1}^2 = \text{argmax}_{\boldsymbol{\theta}^2} V^2(\boldsymbol{\theta}_i^1, \boldsymbol{\theta}^2).$$

In the reinforcement learning setting, agents do not have access to $\{V^1, V^2\}$ over all parameter values. Instead, we assume that agents only have access to the function values and gradients at $(\boldsymbol{\theta}_i^1, \boldsymbol{\theta}_i^2)$. Using this information the naive learn-

ers apply the gradient ascent update rule $\boldsymbol{f}^1_{\text{nl}}$:

$$\begin{aligned}\boldsymbol{\theta}^1_{i+1} &= \boldsymbol{\theta}^1_i + \boldsymbol{f}^1_{\text{nl}}(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i),\\ \boldsymbol{f}^1_{\text{nl}} &= \nabla_{\boldsymbol{\theta}^1_i} V^1(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i) \cdot \delta,\end{aligned} \tag{4.1}$$

where δ is the step size.

4.2 Learning with Opponent Learning Awareness

A LOLA learner optimizes its return under one step look-ahead of opponent learning. Instead of optimizing the expected return under the current parameters, $V^1(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i)$, a LOLA agent optimizes $V^1(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i + \Delta\boldsymbol{\theta}^2_i)$, which is the expected return after the opponent updates its policy with one learning step, $\Delta\boldsymbol{\theta}^2_i$. Going forward we have drop the subscript i for clarity. Assuming small $\Delta\boldsymbol{\theta}^2$, a first-order Taylor expansion results in:

$$V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2 + \Delta\boldsymbol{\theta}^2) \approx V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) + (\Delta\boldsymbol{\theta}^2)^T \nabla_{\boldsymbol{\theta}^2} V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2). \tag{4.2}$$

The LOLA objective (4.2) differs from prior work, e.g., Zhang and Lesser (2010), that predicts the opponent's policy parameter update and learns a best response. LOLA learners attempt to actively influence the opponent's future policy update, and explicitly differentiate through the $\Delta\boldsymbol{\theta}^2$ with respect to $\boldsymbol{\theta}^1$. Since LOLA focuses on this shaping of the learning direction of the opponent, the dependency of $\nabla_{\boldsymbol{\theta}^2} V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)$ on $\boldsymbol{\theta}^1$ is dropped during the backward pass. Investigation of how differentiating through this term would affect the learning outcomes is left for future work.

By substituting the opponent agent's naive learning step:

$$\Delta\boldsymbol{\theta}^2 = \nabla_{\boldsymbol{\theta}^2} V^2(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) \cdot \eta \tag{4.3}$$

into (4.2) and taking the derivative of (4.2) with respect to $\boldsymbol{\theta}^1$, we obtain our LOLA learning rule:

$$\boldsymbol{\theta}^1_{i+1} = \boldsymbol{\theta}^1_i + \boldsymbol{f}^1_{\text{lola}}(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i),$$

which includes a second order correction term

$$\begin{aligned}&\boldsymbol{f}^1_{\text{lola}}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) = \nabla_{\boldsymbol{\theta}^1} V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)\\ &+ \left(\nabla_{\boldsymbol{\theta}^2} V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)\right)^T \nabla_{\boldsymbol{\theta}^1} \nabla_{\boldsymbol{\theta}^2} V^2(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) \cdot \delta\eta,\end{aligned} \tag{4.4}$$

where the step sizes δ, η are for the first and second order updates. LOLA agents can evaluate (4.4) explicitly if they have access to the gradients and Hessians of $\{V^1, V^2\}$ at each agent's current policy parameter $(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_i)$.

4.3 Learning via Policy Gradient

When agents do not have access to exact gradients or Hessians, we derive the update rules $f_{\text{nl, pg}}$ and $f_{\text{lola, pg}}$ based on approximations of the derivatives in (4.1) and (4.4). Denote an episode of horizon T as $\tau = (s_0, u^1_0, u^2_0, r^1_0, r^2_0, ..., s_{T+1}, u^1_T, u^2_T, r^1_T, r^2_T)$ and its corresponding discounted return for agent a at timestep t as $R^a_t(\tau) = \sum_{l=t}^{T} \gamma^{l-t} r^a_l$. Given this definition, the expected episodic return conditioned on the agents' policies (π^1, π^2), $\mathbb{E}\, R^1_0(\tau)$ and $\mathbb{E}\, R^2_0(\tau)$, approximate V^1 and V^2 respectively, as do the gradients and Hessians.

The gradient of $\mathbb{E}\, R^1_0(\tau)$ follows from the policy gradient derivation:

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

where $b(s_t)$ is a baseline for variance reduction. Then the policy gradient-based update rule $\boldsymbol{f}_{\text{nl, pg}}$ for the naive learner is

$$\boldsymbol{f}^1_{\text{nl, pg}} = \nabla_{\boldsymbol{\theta}^1} \mathbb{E}\, R^1_0(\tau) \cdot \delta. \tag{4.5}$$

For the LOLA update, we derive the following estimator of the second-order term in (4.4) based on policy gradients. The derivation (omitted) closely resembles the standard proof of the policy gradient theorem, exploiting that agents sample actions independently. We further note that this second order term is exact in expectation:

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}^1} \nabla_{\boldsymbol{\theta}^2} \mathbb{E}\, R^2_0(\tau)\\ &= \mathbb{E}\left[R^2_0(\tau) \nabla_{\boldsymbol{\theta}^1} \log \pi^1(\tau) \big(\nabla_{\boldsymbol{\theta}^2} \log \pi^2(\tau)\big)^T\right]\\ &= \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r^2_t \cdot \Big(\sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}^1} \log \pi^1(u^1_l|s_l)\Big)\right.\\ &\quad \left.\Big(\sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}^2} \log \pi^2(u^2_l|s_l)\Big)^T\right].\end{aligned} \tag{4.6}$$

The complete LOLA update for agent 1 using policy gradients is

$$\begin{aligned}&\boldsymbol{f}^1_{\text{lola, pg}} = \nabla_{\boldsymbol{\theta}^1} \mathbb{E}\, R^1_0(\tau) \cdot \delta +\\ &\left(\nabla_{\boldsymbol{\theta}^2} \mathbb{E}\, R^1_0(\tau)\right)^T \nabla_{\boldsymbol{\theta}^1} \nabla_{\boldsymbol{\theta}^2} \mathbb{E}\, R^2_0(\tau) \cdot \delta\eta.\end{aligned} \tag{4.7}$$

4.4 LOLA with Opponent Modeling

Both versions (4.4) and (4.7) of LOLA learning assume that each agent has access to the exact parameters of the opponent. However, in adversarial settings the opponent's parameters are typically obscured and have to be inferred from the the opponent's state-action trajectories. Our proposed opponent modeling is similar to behavioral cloning Ross, Gordon, and Bagnell (2011); Bojarski et al. (2016). Instead of accessing agent 2's true policy parameters $\boldsymbol{\theta}^2$, agent 1 models the opponent's behavior with $\hat{\boldsymbol{\theta}}^2$, where $\hat{\boldsymbol{\theta}}^2$ is estimated from agent 2's trajectories using maximum likelihood:

$$\hat{\boldsymbol{\theta}}^2 = \underset{\boldsymbol{\theta}^2}{\operatorname{argmax}} \sum_t \log \pi_{\boldsymbol{\theta}^2}(u^2_t|s_t) \tag{4.8}$$

Then, $\hat{\boldsymbol{\theta}}^2$ replaces $\boldsymbol{\theta}^2$ in the LOLA update rule, both for the exact version (4.4) using the value function and the gradient based approximation (4.7). We compare the performance of LOLA agents with opponent modeling against policy-gradient based LOLA (4.7) in our experiments.

4.5 Higher Order LOLA

By substituting the naive learning rule (4.3) into the LOLA objective (4.2), the LOLA learning rule so far assumes that

the opponent is a naive learner. We call this setting *first-order LOLA*, which corresponds to the first-order learning rule of the opponent agent. However, we can also consider a higher order LOLA agent that assumes the opponent applies a first-order LOLA learning rule, thus replacing (4.3). This leads to third-order derivatives in the learning rule. While the third-order terms are typically difficult to compute using policy gradient due to high variance, when the exact value function is available it is tractable. We examine the benefits of higher-order LOLA in our experiments.

5 Experimental Setup

In this section, we summarize the settings where we compare the learning behavior of NL and LOLA agents. The first setting (Sec. 5.1) consists of two classical infinitely iterated games, the iterated prisoners dilemma (IPD) and iterated matching pennies (IMP). 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, which leads to exact policy updates for NL and LOLA agents. The second setting (Sec. 5.2) is called 'Coin Game', 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. The policy of each player is parameterized with a deep recurrent neural network.

In the policy gradient experiments with LOLA, we assume offline-learning, i.e., agents play many (batch-size) parallel episodes using their latest policies. Policies remain unchanged within each episode, with learning happening between episodes. One setting where this kind of offline learning naturally arises is when training policies using real-world data. E.g., in the case of autonomous cars all data from a fleet of cars can be collected over night and used for training in order to release new policies the next day.

5.1 Iterated Games

We first review the two iterated games, IPD and IMP, and explain how we can model iterated games as memory-1 two-agent MDP.

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

Table 1: Payoff matrix of prisoners' dilemma.

Table 1 shows the per-step payoff matrix of the prisoners' dilemma. In a single-shot prisoners' dilemma, there is only one Nash equilibrium Fudenberg and Tirole (1991), where both agents defect. In the infinitely iterated prisoners' dilemma, the folk theorem (Roger, 1991) shows that there are infinitely many Nash equilibria. Two notable ones are the always defect strategy (DD), and tit-for-tat (TFT). In TFT each agent starts out with cooperation and then repeats the previous action of the opponent. The average returns per step in self-play are -1 and -2 for TFT and DD respectively.

Matching pennies Gibbons (1992) is a zero-sum game, with per-step payouts shown in Table 2. This game only has a single mixed strategy Nash equilibrium which is both players playing $50\%/50\%$ heads / tails.

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

Table 2: Payoff matrix of matching pennies.

Agents in both IPD and IMP can condition their actions on past history. Agents in an iterated game are endowed with a memory of length K if the agents act based on the results of the last K rounds. Press and Dyson Press and Dyson (2012) proved that agents with a good memory-1 strategy can effectively force the iterated game to be played as memory-1. Thus, we consider memory-1 iterated games in our work.

We can model the memory-1 IPD and IMP as a two-agent MDP, where the state at time 0 is empty, denoted as s_0, and at time $t \geq 1$ is both agents' actions from $t-1$:

$$s_t = (u^1_{t-1}, u^2_{t=1}) \quad \text{for } t > 1.$$

Each agent's policy is fully parametrized by 5 probabilities. For agent a in the case of the IPD, they are the probability of cooperation at game start $\pi^a(C|s_0)$, and the cooperation probabilities in the four memories: $\pi^a(C|CC)$, $\pi^a(C|CD)$, $\pi^a(C|DC)$, and $\pi^a(C|DD)$. By analytically solving the multi-agent MDP we can derive each agent's future discounted reward as an analytical function of the agents' policies and calculate the exact policy update for both NL and LOLA agents.

We also organize a round-robin tournament where we compare LOLA-Ex to a number of state of the art multi-agent learning algorithms, both on the IPD and IMP.

5.2 Coin Game

Next, we study LOLA in a more high-dimensional setting called the 'Coin Game', where each round requires agents to take sequential actions and we parametrize agents' policies with deep neural networks. The 'Coin Game' was first

	IPD		IMP	
	%TFT	R(std)	%Nash	R(std)
NL-Ex.	20.8	-1.98(0.14)	0.0	0(0.37)
LOLA-Ex.	81.0	-1.06(0.19)	98.8	0(0.02)
NL-PG	20.0	-1.98(0.00)	13.2	0(0.19)
LOLA-PG	66.4	-1.17(0.34)	93.2	0(0.06)

Table 3: We summarize results for NL vs. NL and LOLA vs. LOLA settings with either exact gradient evaluation or policy gradient approximation. Shown is the probability of agents playing TFT and Nash for the IPD and IMP respectively as well as the average reward per step, R, and (STD) at the end of training for 50 training runs.

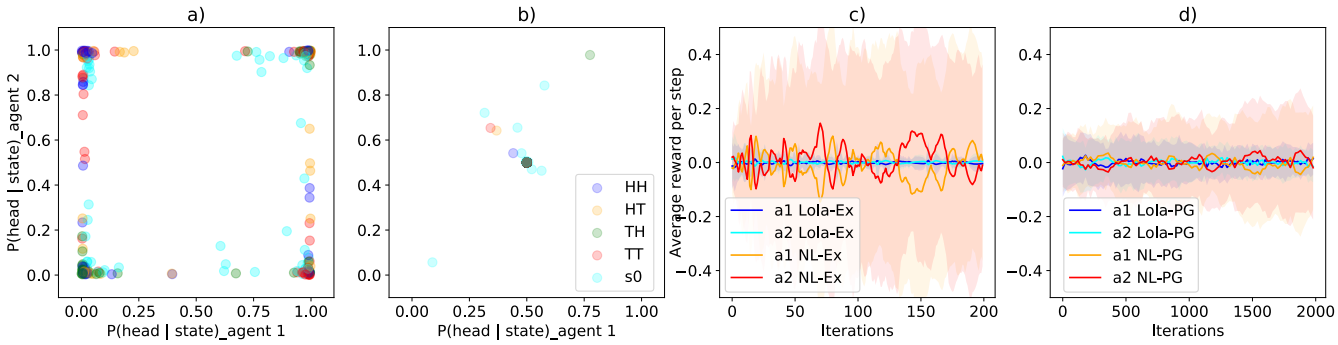


Figure 2: Shown is the probability of playing heads in the iterated matching pennies (IMP) at the end of 50 training runs for both agents as a function of state under naive learning NL-Ex, a), and LOLA-Ex b) when using the exact gradients of the value function. Also shown is the normalized discounted return for both agents in NL-Ex vs. NL-Ex and LOLA-Ex vs. LOLA-Ex with exact gradient, c), and the normalized discounted return for both agents in NL-PG vs. NL-PG and LOLA-PG vs. LOLA-PG with policy gradient approximation, d). We can see in a) that NL-Ex results in near deterministic strategies, indicated by the accumulation of points in the corners. These strategies are easily exploitable by other deterministic strategies leading to unstable training and high variance in the reward per step in c). In contrast, LOLA agents learn to play the only Nash strategy, $50\%/\%50$, leading to low variance in the reward per step. One interpretation is that LOLA agents anticipate that exploiting a deviation from Nash increases their immediate return, but also renders them more exploitable by the opponent's next learning step. Best viewed in color.

proposed in Lerer and Peysakhovich (2017) as a higher dimensional expansion of the iterated prisoner's dilemma with multi-step actions. As shown in Figure 1, in this setting two agents, 'red' and 'blue', are tasked with collecting coins.

The coins are either blue or red, and appear randomly on the grid-world. A new coin with random color and random position appears after the last one is picked up. Agents pick up coins by moving onto the position where the coin is located. While every agent receives a point for picking up a coin of any colour, whenever an picks up a coin of different color, the other agent loses 2 points.

As a result, if both agents greedily pick up any coin available, they receive 0 points in expectation. In 'Coin Game', agents' policies are parametrized with a recurrent neural network and one cannot obtain the future discounted reward as a function of both agents' policies in closed form. Policy gradient-based learning is applied for both NL and LOLA agents in our experiments. We further experiment LOLA with opponent-modelling in this environment to examine the behavior of LOLA agents without access to the opponent's policy parameters.

5.3 Training Details

In policy gradient-based NL and LOLA settings, we train agents with actor-critic method (Sutton and Barto, 1998) and parametrize each agent with a policy actor, and a policy critic for variance reduction during policy updates.

During training, we use gradient descent with step size 0.005 for the actor, 1 for the critic, and the batch size 4000 for rollouts. The discout rate γ is set to 0.96 for the prisoners' dilemma and the coin game and 0.9 for matching pennies. The high value of γ for the 'Coin Game' and IPD was chosen in order to allow for long time horizons, which are known to be required for cooperation in the IPD. We found that a lower γ produced more stable learning on the IMP.

For the coin game the agent's policy architecture is a recurrent neural network with 32 hidden units and 2 convolutional layers with 3×3 filters, stride 1, and 'relu' activation for input processing. The input is presented as a 4 channel grid, with 2 channels encoding the positions of the 2 agents and 2 channels for the red and blue coins respectively.

For the tournament, we use baseline algorithms and the corresponding hyperparameter values as provided in the literature (Bowling and Veloso, 2002). The tournament is played in a round-robin fashion between all pairs of agents for 1000 episodes, 200 steps each.

6 Results

In this section, we summarize the experimental results. We aim to answer the following questions:

1. With the exact policy update, how do pairs of LOLA agents behave in iterated games compared with pairs of NL agents?
2. How do LOLA-Ex agents fair in a round robin tournament involving a set of multi-agent learning algorithms from literature?
3. Does replacing the exact policy update with policy gradient updates change the learned behaviors of LOLA and NL agents?
4. Does the learning of LOLA agents scale to high-dimensional settings where the agents' policies are parametrized by deep neural networks?
5. When replacing access to the exact parameters of the opponent agent with opponent modeling, does LOLA

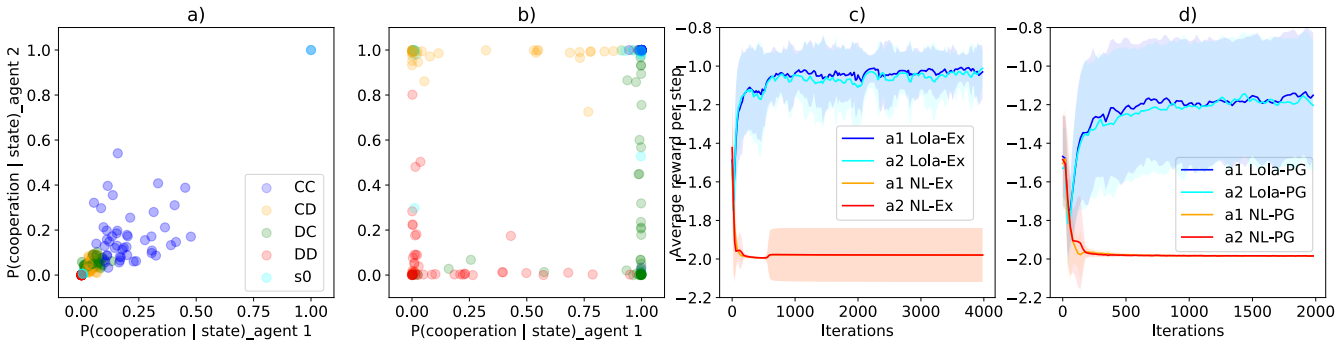


Figure 3: Shown is the probability of cooperation in the iterated prisoners dilemma (IPD) at the end of 50 training runs for both agents as a function of state under naive learning NL-Ex a), and LOLA-Ex b) when using the exact gradients of the value function. Also shown is the normalized discounted return for both agents in NL-Ex vs. NL-Ex and LOLA-Ex vs. LOLA-Ex, with the exact gradient, c), and the normalized discounted return for both agents in NL-PG vs. NL-PG and LOLA-PG vs. LOLA-PG, with policy gradient approximation, d). We can see that NL-Ex leads to DD, resulting in an average reward of ca. -2. In contrast, the LOLA-Ex agents play tit-for-tat in b): When in the last move agent 1 defected and agent 2 cooperated (DC, green points), most likely in the next move agent 1 will cooperate and agent 2 will defect, indicated by a concentration of the green points in the bottom right corner. Similarly, the yellow points (CD), are concentrated in the top left corner. While the results for the NL-PG and LOLA-PG with policy gradient approximation are more noisy, they are qualitatively similar. Best viewed in color.

agents' behavior preserve?

6. Exploiting LOLA: Can LOLA agents be exploited by using higher order gradients, i.e., does LOLA lead to an arms race of ever higher order corrections or is LOLA / LOLA stable?

We answer the first two questions in Sec. 6.1, the next two questions in Sec. 6.2 and the last one in Sec. 6.3.

6.1 Iterated Games

We first compare the behaviors of LOLA agents with NL agents, with either exact policy updates or policy gradient updates.

Figures 3a) and 3b) show the policy for both agents at the end of training under naive learning (NL-Ex) and LOLA (LOLA-Ex) when the agents have access to exact gradients and Hessians of $\{V^1, V^2\}$. Here we consider the settings of NL vs NL and LOLA vs LOLA. We study mixed learning of one LOLA agent vs. an NL agent in Section 6.3. Under NL-Ex, the agents learn to defect in all states, indicated by the accumulation of points in the bottom left corner of the plot. However, under LOLA-Ex, in most cases the agents learn TFT. In particular agent 1 cooperates in the starting state s_0, CC and DC, while agent 2 cooperates in s_0, CC and CD. As a result, Figure 3c) shows that the normalized discounted reward[1] is close to -1 for LOLA-Ex vs. LOLA-Ex, corresponding to TFT, while NL vs. NL results in an normalized discounted reward of -2, corresponding to the fully defective (DD) equilibrium. Figure 3d) shows the normalized discounted reward for NL-PG and LOLA-PG where agents learn via policy gradient. LOLA-PG also demonstrates cooperation while agents defect in NL-PG.

We conduct the same analysis for IMP in Figure 2. In this game, under naive learning the agents' strategies fail to converge. In contrast, under LOLA the agents' policies converge to the only Nash equilibrium, playing $50\%/50\%$ heads / tails.

Table 3 summarizes the numerical results comparing LOLA with NL agents in both the exact and policy gradient settings in the two iterated game environments. In IPD, LOLA agents learn policies consistent with TFT with a much higher probability and achieve higher normalized discounted rewards than NL (-1.06 vs -1.98). In IMP, LOLA agents converge to the Nash equilibrium more stably while NL agents do not. The difference in stability is illustrated by the high variance of the normalized discounted returns for NL agents compared to the low variance under LOLA (0.37 vs 0.02).

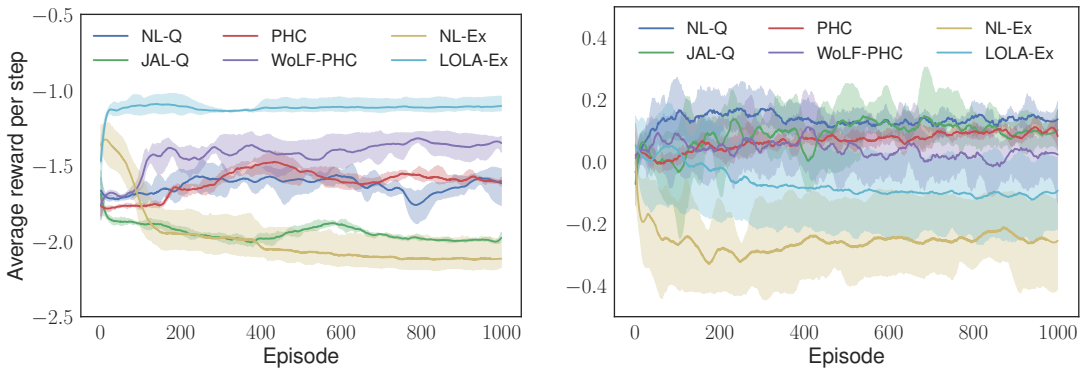
In Figure 4 we show the average normalized return of our LOLA-Ex agent against a set of learning algorithms from literature. We find that LOLA-Ex receives the highest normalized return in the IPD, clearly indicating that it successfully shapes the learning outcome of other algorithms in this general sum setting.

In IMP LOLA-Ex achieves stable performance close to the middle of the distribution of results.

6.2 Coin Game

We summarize our experiment results in the Coin Game environment. To examine the scalability of LOLA learning rules, we compare NL-PG vs. NL-PG and LOLA-PG vs. LOLA-PG. Figure 5 demonstrates that NL-PG agents collect coins indiscriminately, corresponding to defection. In

[1] We use following definition for the normalized discounted reward: $(1-\gamma)\sum_{t=0}^{T}\gamma^t r_t$.



(a) (b)

Figure 4: Shown are the normalized returns of a round-robin tournament on IPD in a) and IMP in b). LOLA-Ex agents achieve the best performance on IPD and are within error bars for IMP. Shading indicates a 95% confidence interval of the error of the mean. Baselines from (Bowling and Veloso, 2002): naive Q-learner (NL-Q), joint-action Q-learner (JAL-Q), policy hill-climbing (PHC), and "Win or Learn Fast" (WoLF) PHC.

contrast, LOLA-PG agents learn to pick up coins predominantly (around 80%) of their own color, showing that LOLA learning rule leads to cooperation Coin Game as well.

Removing the assumption that agents can access the exact parameters of opponents, we examine LOLA agents with opponent modeling (Section 4.4). Figure 5 demonstrates that without access to the opponent's policy parameters, LOLA agents with opponent modeling pick up coins of their own color around 70% of the time, slightly inferior compared to the performance of LOLA-PG agents. We emphasize that with opponent modeling neither agent can recover the exact policy parameters of the opponent, since there is a large amount of redundancy in the neural network parameters. For example, each agent could permute the weights of their fully connected layers. Opponent modeling introduces noise in the opponent agent's policy parameters, thus increasing the variance of the gradients (4.7) during policy updates, which leads to inferior performance of LOLA-OM vs. LOLA-PG in Figure 5.

6.3 Exploitability of LOLA

We address the exploitability of LOLA learning rule in this section. We consider the IPD setting, where one can calculate the exact value function of each agent giving their policies. Thus, we can evaluate the higher order LOLA terms. We pitch a NL-Ex or LOLA-Ex agent against NL-Ex, LOLA-Ex, and a 2nd-order LOLA agent. We compare the normalized discounted return of each agent in all settings and address the question of whether there is an arms race to incorporate ever higher orders of LOLA correction terms between the two agents.

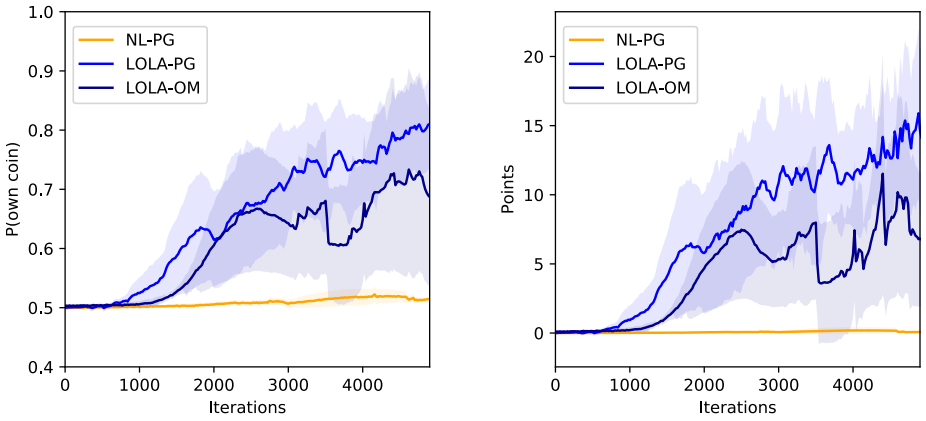
Table 4 shows a LOLA-Ex learner can achieve higher payouts against NL-Ex. Thus, there is an incentive for either agent to switch from naive learning to first order LOLA. Furthermore, two LOLA-Ex agents playing against each other both receive higher normalized discounted reward than a LOLA-Ex agent playing against a NL-Ex. This makes LOLA a dominant learning rule in IPD compared to naive learning. However, we further find that 2nd-order LOLA provides no incremental gains when playing against a LOLA-Ex agent, leading to a reduction in payouts for both agents. These experiments were carried out with a LR of 0.5. While it is beyond the scope of this work to prove that LOLA vs LOLA is a dominant learning rule in the space of all possible gradient-based rules, these initial results are encouraging.

	NL-Ex	LOLA-Ex	2nd-Order
NL-Ex	(-1.99, -1.99)	(-1.54, -1.28)	-
LOLA-Ex	(-1.28, -1.54)	(-1.04, -1.04)	(-1.14, -1.17)

Table 4: Higher order LOLA results on the IPD. A LOLA-Ex agent obtains higher normalized return compared to a NL-Ex agent. However in this setting there is no incremental gain from using higher order LOLA in order to exploit another LOLA agent in the IPD. In fact both agents do worse with the 2nd order corrections.

7 Conclusions & Future Work

We presented Learning with Opponent-Learning Awareness (LOLA), a learning method for multi-agent settings that considers the learning processes of other agents. We show that when both agents have access to exact value function and apply the LOLA learning rule, cooperation emerges based on tit-for-tat in the infinitely repeated iterated prisoners' dilemma while independent naive learners defect. We also find that LOLA leads to stable learning of the Nash equilibrium in IMP. In our round-robin tournament against other multi-agent learning algorithms we show that exact LOLA agents achieve the highest average returns on the IPD and respectable performance on IMP. We also derive a policy gradient-based version of LOLA, applicable to deep reinforcement learning setting. Experiments on IPD and IMP



(a) (b)

Figure 5: Shown is the percentage of all picked up coins that match in colour, in a), and the total points obtained, in b), for a pair of naive learners using policy gradient (NL-PG), LOLA-agents (LOLA-PG), and a pair of LOLA-agents with opponent modelling (LOLA-OM). Also shown is the standard deviation of the percentage and the points obtained in order to indicate variability of the result, based on 5 training runs. We see that LOLA and LOLA-OM learn to cooperate, while NL does not. Best viewed in color.

demonstrate similar learning behavior to the setting with exact value function.

In addition, we scale the policy gradient-based version of LOLA to 'Coin Game', a multi-step game which requires deep recurrent policies. LOLA agents learn to cooperate, as colored agents pick up coins of their color with high probability while naive learners pick up coins indiscriminately. We further remove agents' access to the opponent agents' policy parameters and replace with opponent modeling. LOLA agents with opponent modeling also learn to cooperate.

We briefly address the exploitability of LOLA agents. Empirical results show that in the IPD both agents are incentivized to use LOLA, while higher order exploits show no further gain.

In the future we would like to continue to address the exploitability of LOLA, when adversarial agents explicitly aim to take advantage of a LOLA learner using global search methods rather than using gradient-based methods only. Just as LOLA is a way to exploit a naive learner, there should be means of exploiting LOLA learners in turn, unless LOLA is itself an equilibrium learning strategy.

Acknowledgements

We would like to thank Jascha Sohl-Dickstein, David Balduzzi, Karl Tuyls, Marc Lanctot, Michael Bowling, Ilya Sutskever, Bob McGrew, and Paul Cristiano for fruitful discussion. We would like to thank Michael Littman for providing feedback on an early version of the manuscript. We would like to thank our reviewers for critical and thoughtful feedback.

References

Axelrod, R. M. 2006. *The evolution of cooperation: revised edition*. Basic books.

Banerjee, D., and Sen, S. 2007. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems* 15(1):91–108.

Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L. D.; Monfort, M.; Muller, U.; Zhang, J.; et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bowling, M., and Veloso, M. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence* 136(2):215–250.

Brafman, R. I., and Tennenholtz, M. 2003. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems*, volume 9, 1635–1643.

Brown, G. W. 1951. Iterative solution of games by fictitious play.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*.

Claus, C., and Boutilier, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998:746–752.

Crandall, J. W., and Goodrich, M. A. 2011. Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Machine Learning* 82(3):281–314.

Das, A.; Kottur, S.; Moura, J. M.; Lee, S.; and Batra, D. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

Foerster, J.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Foerster, J.; Nardelli, N.; Farquhar, G.; Torr, P.; Kohli, P.; Whiteson, S.; et al. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *34th International Conference of Machine Learning*.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *AAAI*.

Fudenberg, D., and Tirole, J. 1991. Game theory, 1991. *Cambridge, Massachusetts* 393:12.

Gibbons, R. 1992. *Game theory for applied economists*. Princeton University Press.

Heinrich, J., and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

Hernandez-Leal, P., and Kaisers, M. 2017. Learning against sequential opponents in repeated stochastic games.

Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; and de Cote, E. M. 2017. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.

Kleiman-Weiner, M.; Ho, M. K.; Austerweil, J. L.; Littman, M. L.; and Tenenbaum, J. B. 2016. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *COGSCI*.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Perolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2016. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 464–473. International Foundation for Autonomous Agents and Multiagent Systems.

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

Littman, M. L. 2001. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, 322–328.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

Mealing, R., and Shapiro, J. L. 2013. Opponent modelling by sequence prediction and lookahead in two-player games. In *ICAISC (2)*, 385–396.

Mealing, R., and Shapiro, J. 2015. Opponent modelling by expectation-maximisation and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games*.

Metz, L.; Poole, B.; Pfau, D.; and Sohl-Dickstein, J. 2016. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*.

Mordatch, I., and Abbeel, P. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Munoz de Cote, E., and Littman, M. L. 2008. A polynomial-time Nash equilibrium algorithm for repeated stochastic games. In *24th Conference on Uncertainty in Artificial Intelligence (UAI'08)*.

Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent rl under partial observability. *arXiv preprint arXiv:1703.06182*.

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

Roger, B. M. 1991. Game theory: analysis of conflict.

Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer.

Sandholm, T. W., and Crites, R. H. 1996. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems* 37(1-2):147–166.

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2244–2252.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, 1057–1063.

Tuyls, K.; Heytens, D.; Nowe, A.; and Manderick, B. 2003. Extended replicator dynamics as a key to reinforcement learning in multi-agent systems. In *European Conference on Machine Learning*, 421–431. Springer.

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

Wunder, M.; Littman, M. L.; and Babes, M. 2010. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1167–1174.

Zawadzki, E.; Lipson, A.; and Leyton-Brown, K. 2014. Empirically evaluating multiagent learning algorithms. *arXiv preprint arXiv:1401.8074*.

Zhang, C., and Lesser, V. R. 2010. Multi-agent learning with policy prediction. In *AAAI*.

Zinkevich, M.; Greenwald, A.; and Littman, M. L. 2006. Cyclic equilibria in markov games. In *Advances in Neural Information Processing Systems*, 1641–1648.

A Appendix

A.1 Derivation of Second-Order derivative

In this section, we derive the second order derivatives of LOLA in the policy gradient setting. Recall that an episode of horizon T is

$$\tau = (s_0, u_0^1, u_0^2, r_0^1, r_0^2, ..., s_T, u_T^1, u_T^2, r_T^1, r_T^2)$$

and the corresponding discounted return for agent a at timestep t is $R_t^a(\tau) = \sum_{l=t}^{T} \gamma^{l-t} r_l^a$. We denote $\mathbb{E}_{\pi^1,\pi^2,\tau}$ as the expectation taken over both agents' policy and the episode τ. Then,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}^1}\nabla_{\boldsymbol{\theta}^2}\,\mathbb{E}_{\pi^1,\pi^2,\tau}\,R_0^1(\tau) &= \nabla_{\boldsymbol{\theta}^1}\nabla_{\boldsymbol{\theta}^2}\,\mathbb{E}_{\tau}\left[R_0^1(\tau)\cdot\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\cdot\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\right] \\
&= \mathbb{E}_{\tau}\left[R_0^1(\tau)\cdot\left(\nabla_{\boldsymbol{\theta}^1}\big(\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\big)\right)\left(\nabla_{\boldsymbol{\theta}^2}\big(\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\big)\right)^T\right] \\
&= \mathbb{E}_{\tau}\left[R_0^1(\tau)\cdot\left(\frac{\nabla_{\boldsymbol{\theta}^1}\big(\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\big)}{\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)}\right)\left(\frac{\nabla_{\boldsymbol{\theta}^2}\big(\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\big)}{\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)}\right)^T\right. \\
&\quad\left.\cdot\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\cdot\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\right] \\
&= \mathbb{E}_{\pi^1,\pi^2,\tau}\left[R_0^1(\tau)\cdot\left(\frac{\nabla_{\boldsymbol{\theta}^1}\big(\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\big)}{\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)}\right)\left(\frac{\nabla_{\boldsymbol{\theta}^2}\big(\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\big)}{\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)}\right)^T\right] \\
&= \mathbb{E}_{\pi^1,\pi^2,\tau}\left[R_0^1(\tau)\cdot\left(\nabla_{\boldsymbol{\theta}^1}\log\big(\prod_{l=0}^{T}\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\big)\right)\left(\nabla_{\boldsymbol{\theta}^2}\log\big(\prod_{l=0}^{T}\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\big)\right)^T\right] \\
&= \mathbb{E}_{\pi^1,\pi^2,\tau}\left[R_0^1(\tau)\cdot\left(\sum_{l=0}^{T}\nabla_{\boldsymbol{\theta}^1}\log\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\right)\left(\sum_{l=0}^{T}\nabla_{\boldsymbol{\theta}^2}\log\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\right)^T\right].
\end{aligned}$$

The second equality is due to π_l is only a function of $\boldsymbol{\theta}_l$. The third equality is multiply and divide the probability of the episode τ. The fourth equality factors the probability of the episode τ into the expectation $\mathbb{E}_{\pi^1,\pi^2,\tau}$. The fifth and sixth equalities are standard policy gradient operations.

Similar derivations lead to the the following second order cross-term gradient for a single reward of agent 1 at time t

$$\nabla_{\boldsymbol{\theta}^1}\nabla_{\boldsymbol{\theta}^2}\,\mathbb{E}_{\pi^1,\pi^2,\tau}\,r_t^1 = \mathbb{E}_{\pi^1,\pi^2,\tau}\left[r_t^1\cdot\left(\sum_{l=0}^{t}\nabla_{\boldsymbol{\theta}^1}\log\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\right)\left(\sum_{l=0}^{t}\nabla_{\boldsymbol{\theta}^2}\log\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\right)^T\right].$$

Sum the rewards over t,

$$\nabla_{\boldsymbol{\theta}^1}\nabla_{\boldsymbol{\theta}^2}\,\mathbb{E}_{\pi^1,\pi^2,\tau}\,R_0^1(\tau) = \mathbb{E}_{\pi^1,\pi^2,\tau}\left[\sum_{t=0}^{T}\gamma^t r_t^1\cdot\left(\sum_{l=0}^{t}\nabla_{\boldsymbol{\theta}^1}\log\pi^1(u_l^1|s_l,\boldsymbol{\theta}^1)\right)\left(\sum_{l=0}^{t}\nabla_{\boldsymbol{\theta}^2}\log\pi^2(u_l^2|s_l,\boldsymbol{\theta}^2)\right)^T\right],$$

which is the 2nd order term in the Methods Section.

A.2 Derivation of the exact value function in the Iterated Prisoners' dilemma and Iterated Matching Pennies

In both IPD and IMP the action space consists of 2 discrete actions. The state consists of the union of the last action of both agents. As such there are a total of 5 possible states, 1 state being the initial state, s_0, and the other 4 the 2 x 2 states depending on the last action taken.

As a consequence the policy of each agent can be represented by 5 parameters,$\boldsymbol{\theta}^a$,the probabilities of taking action 0 in each of these 5 states. In the case of the IPD these parameters correspond to the probability of cooperation in s_0, CC, CD, DC and

DD:

$$\begin{aligned}
\pi^a(C|s_0) &= \theta^{a,0}, \quad \pi^a(D|s_0) = 1 - \theta^{a,0}, \\
\pi^a(C|CC) &= \theta^{a,1}, \quad \pi^a(D|CC) = 1 - \theta^{a,1}, \\
\pi^a(C|CD) &= \theta^{a,2}, \quad \pi^a(D|CD) = 1 - \theta^{a,2}, \\
\pi^a(C|DC) &= \theta^{a,3}, \quad \pi^a(D|DC) = 1 - \theta^{a,3}, \\
\pi^a(C|DD) &= \theta^{a,4}, \quad \pi^a(D|DD) = 1 - \theta^{a,4}, \quad a \in \{1, 2\}.
\end{aligned}$$

We denote $\boldsymbol{\theta}^a = (\theta^{a,0}, \theta^{a,1}, \theta^{a,2}, \theta^{a,3}, \theta^{a,4})$.

In these games the union of π^1 and π^2 induces a state transition function $P(s'|s) = P(\boldsymbol{u}|s)$. Denote the distribution of s_0 as $\boldsymbol{p}_0$:

$$\boldsymbol{p}_0 = \left(\theta^{1,0}\theta^{2,0},\ \theta^{1,0}(1-\theta^{2,0}),\ (1-\theta^{1,0})\theta^{2,0},\ (1-\theta^{1,0})(1-\theta^{2,0})\right)^T,$$

the payout vector as

$$\boldsymbol{r}^1 = (-1, -3, 0, -2)^T \quad \text{and} \quad \boldsymbol{r}^2 = (-1, 0, -3, -2)^T,$$

and the transition matrix is

$$\boldsymbol{P} = \left[\boldsymbol{\theta}^1\boldsymbol{\theta}^2, \quad \boldsymbol{\theta}^1(\mathbf{1} - \boldsymbol{\theta}^2), \quad (\boldsymbol{\theta}^1 - \mathbf{1})\boldsymbol{\theta}^2, \quad (\mathbf{1} - \boldsymbol{\theta}^1)(\mathbf{1} - \boldsymbol{\theta}^2)\right]$$

Then V_1, V_2 can be represented as

$$\begin{aligned}
V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) &= \boldsymbol{p}_0^T\left(\boldsymbol{r}^1 + \sum\nolimits_{t=1}^{\infty} \gamma^t \boldsymbol{P}^t \boldsymbol{r}^1\right) \\
V^2(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) &= \boldsymbol{p}_0^T\left(\boldsymbol{r}^2 + \sum\nolimits_{t=1}^{\infty} \gamma^t \boldsymbol{P}^t \boldsymbol{r}^2\right).
\end{aligned}$$

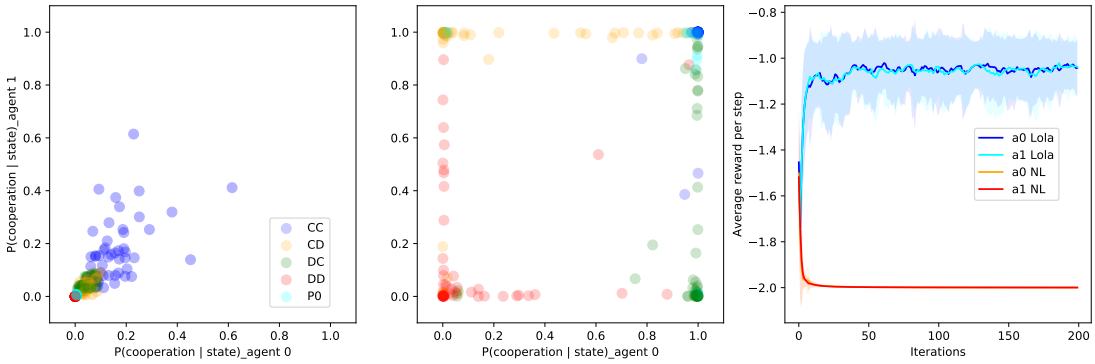
Since $\gamma < 1$ and $\boldsymbol{P}$ is a stochastic matrix, the infinite sum converges and

$$\begin{aligned}
V^1(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) &= \boldsymbol{p}_0^T \frac{\mathbf{I}}{\mathbf{I} - \gamma \boldsymbol{P}} \boldsymbol{r}^1, \\
V^2(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) &= \boldsymbol{p}_0^T \frac{\mathbf{I}}{\mathbf{I} - \gamma \boldsymbol{P}} \boldsymbol{r}^2,
\end{aligned}$$

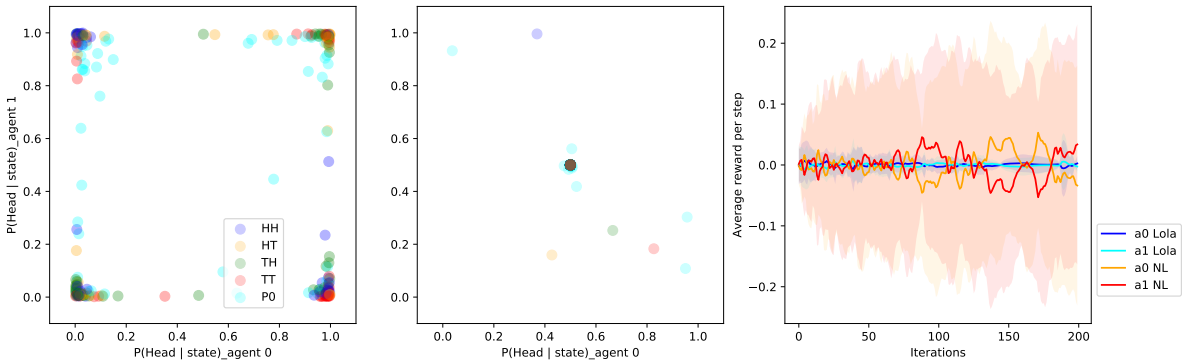
where $\mathbf{I}$ is the identity matrix.

An equivalent derivation holds for the Iterated Matching Pennies game with $\boldsymbol{r}^1 = (-1, 1, 1, -1)^T$ and $\boldsymbol{r}^2 = -\boldsymbol{r}^1$.

A.3 Figures

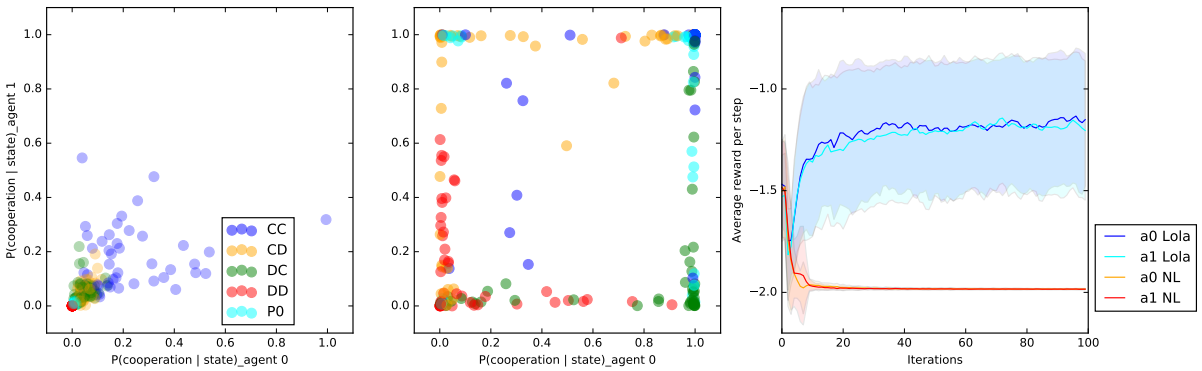


(a)

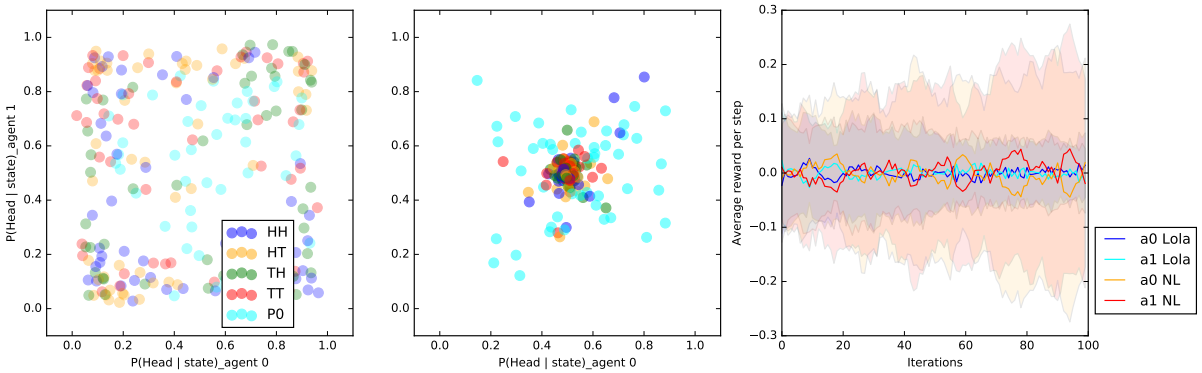


(b)

Figure 6: Shown is the probability of cooperation in the prisoners dilemma (a) and the probability of heads in the matching pennies game (b) at the end of 50 training runs for both agents as a function of state under naive learning (left) and LOLA (middle) when using the exact gradients of the value function. Also shown is the average return per step for naive and LOLA (right)



(a)



(b)

Figure 7: Same as Figure A.3, but using the policy gradient approximation for all terms. Clearly results are more noisy by qualitatively follow the results of the exact method.