Learning with Opponent-Learning Awareness

Jakob N. Foerster²† javo, foerster@cs.ox.ac.uk  Richard Y. Chen¹† richardchen@openai.com  Maruan Al-Shedivat⁴ alshedivat@cs.cmu.edu
Shimon Whiteson² shimon.whiteson@cs.ox.ac.uk  Pieter Abbeel¹,3 pieter@openai.com  Igor Mordatch¹ mordatch@openai.com

¹OpenAI  ²University of Oxford  ³UC Berkeley  ⁴CMU

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, only LOLA agents converge to the Nash equilibrium. 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., 2017a,b] 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 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 ‘opponent’ 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.

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 policy gradients, a standard 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 modeling 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), 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.

Most work in deep multi-agent RL focuses on fully cooperative settings (Omidshafiei et al., 2017; Foerster et al., 2017a,b) and emergent communication in particular (Das et al., 2017; Mordatch and Abbeel, 2017; Lazaridou, Peysakhovich, and Barrett, 2016; Foerster et al., 2016; Sukhbaatar, Fergus, and others, 2016). As an exception, Leibo et al., (2017) consider mixed multi-agent environments and study the emergence of cooperation and competition as a function of the problem setup and the model parameters. Similarly, Lowe et al., (2017) propose a centralized actor-critic architecture for efficient training in these mixed environments. However, neither of these papers explicitly reason about the learning behaviour of other agents and thus fails to discover interesting solutions in mixed-competitive settings.

The closest to our problem setting is the work of Lerer and Peysakhovich (2017), which directly generalizes 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.

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

Brafman and Tenenboim (2003) introduce the concept of an ‘efficient learning equilibrium’ (ELE), in which neither side is encouraged to deviate from the learning rule. Their algorithm applies to settings where all Nash equilibria can be computed and enumerated. So far no proof exists that LOLA is an ELE but our initial empirical results are encouraging. Furthermore, we do not assume that Nash equilibria are computable, which is in general difficult in high-dimensional complex settings. For example, listing all the Nash equilibria of the board game Go is clearly beyond the scope of current techniques.

Our work also relates to opponent modeling, such as fictitious play (Brown, 1951) and action prediction. Mealig and Shapiro (2013) propose a method that finds a policy based on predicting the opponent’s future action. 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.

With LOLA, each agent differentiates its estimated reward 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 a end-to-end differentiable loss function, and thus does not work in the general RL setting. However, the overall results are similar: anticipating the opponents update stabilises the training outcome.

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, $a \in A = \{1, \ldots, n\}$, choose actions, $u^a \in U^a$, and $s$ is the state of the environment. The joint action $u \in U^a \equiv U^n$ leads to a state transition based on the transition function $P(s' | s, u) : S \times U \times S \to [0, 1]$. The reward functions $r^a(s, u) : S \times U \to \mathbb{R}$ 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 $R^a_t = \sum_{t=0}^{\infty} \gamma^t r^a_{t+t}$ for each agent, $a$. As a naive learner, each agent maximizes its total discounted re-
Suppose each agent’s policy \( \pi^a \) is parameterized by \( \theta^a \) and \( V^a(\theta^1, \theta^2) \) is the expected total discounted return for agent \( a \) as a function of both agents’ policy parameters \( (\theta^1, \theta^2) \). A naive learner iteratively optimizes for its own expected total discounted return separately, such that at the \( i \)th iteration, \( \theta^a_i \) is updated to \( \theta^a_{i+1} \) according to

\[
\theta^a_{i+1} = \arg\max_{\theta^a} V^a(\theta^1_i, \theta^2_i),
\]

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 \( (\theta^1, \theta^2) \). Using this information the naive learners apply the gradient ascent update rule \( f^a_{i+1} \):

\[
\theta^a_{i+1} = \theta^a_i + f^a_{i+1}(\theta^1_i, \theta^2_i),
\]

where \( \delta \) is the step size.

4.2 Learning with Opponent Learning Awareness

A LOLA learner optimizes its policy by driving the opponent’s best-response policy to maximize its own expected discounted future return, such that in every iteration,

\[
\theta^a_{i+1} = \theta^a_i + \Delta \theta^a, \quad a \in \{0, 1\},
\]

where

\[
\Delta \theta^1 = \arg\max_{\Delta \theta^1 : \|\Delta \theta^1\| \leq \delta} V^1(\theta^1_i + \Delta \theta^1),
\]

\[
\Delta \theta^2 = \arg\max_{\Delta \theta^2 : \|\Delta \theta^2\| \leq \delta} V^2(\theta^1_i + \Delta \theta^1, \theta^2_i + \Delta \theta^2)),
\]

\[
\Delta \theta^2 = \arg\max_{\Delta \theta^2 : \|\Delta \theta^2\| \leq \delta} \arg\max_{\Delta \theta^1 : \|\Delta \theta^1\| \leq \delta} V^1(\theta^1_i + \Delta \theta^1, \theta^2_i + \Delta \theta^2, \theta^2_i + \Delta \theta^2).\]

In contrast to prior works, e.g., Zhang and Lesser (2010), that only predict the opponent’s policy parameter update, LOLA learners actively influence the opponent’s future policy update.

If the agents have access only to the gradients and Hessians of \( \{V^1, V^2\} \) at each agent’s current policy parameter
(θ₁, θ₂), then the LOLA update rule \( f_{lola}^a \) augments the gradient ascent of \( f_{\theta}^a \) with a second-order term such that at the \( t \)th iteration, agent 1 updates \( \theta_1 \) to \( \theta_1^{t+1} \) as follows

\[
\theta_1^{t+1} = \theta_1^t + f_{lola}^1(\theta_1, \theta_2^t)
\]

and

\[
f_{lola}^1(\theta_1^t, \theta_2^t) = \frac{\partial^2 V^1(\theta_1^t, \theta_2^t)}{\partial \theta_1^t \partial \theta_2^t} \cdot \delta
\]

\[
+ \left( \frac{\partial V^1(\theta_1^t, \theta_2^t)}{\partial \theta_1^t} \right)^2 \frac{\partial^2 V^2(\theta_1^t, \theta_2^t)}{\partial \theta_1^t \partial \theta_2^t} \cdot \delta \eta,
\]

where the step sizes \( \delta, \eta \) are for the first and second order updates.

### 4.3 Learning via Policy Gradient

When agents do not have access to exact gradients or Hessians, we derive the update rules \( f_{lola, pg} \) and \( f_{lola, pg}^a \) based on approximations of the derivatives in (4.1) and (4.2). Denote an episode of horizon \( T \) as \( \tau = (s_0, u_0^1, u_0^2, r_0^1, r_0^2, \ldots, s_T, u_T^1, u_T^2, r_T^1, r_T^2) \) and its corresponding discounted return for agent \( a \) at timestep \( t \) as \( R_a^t(\tau) = \sum_{i=t}^{T} \gamma^{t-i} r_i^a \). Then the expected episodic return given the agents’ policies \( (\pi^1, \pi^2) \), \( E[R_a^0(\tau)] \) and \( E[R_a^0(\tau)] \), approximate \( V^1 \) and \( V^2 \) respectively, so do the gradients and Hessians.

The gradient of \( E[R_0^1(\tau)] \) follows from the policy gradient derivation:

\[
\nabla_{\theta^1} E[R_0^1(\tau)] = \mathbb{E} \left[ \frac{\partial}{\partial \theta^1} \log \pi^1(\tau) \right] = \mathbb{E} \left[ \sum_{t=0}^{T} \nabla_{\theta^1} \log \pi^1(u_t^1|s_t) \sum_{t=0}^{T} \gamma^t \right]
\]

\[
= \mathbb{E} \left[ \sum_{t=0}^{T} \nabla_{\theta^1} \log \pi^1(u_t^1|s_t) \gamma^t (R_1^t(\tau) - b(s_t)) \right],
\]

where \( b(s_t) \) is a baseline for variance reduction. Then the policy gradient-based update rule \( f_{lola, pg}^a \) for the naive learner is

\[
f_{lola, pg}^a = \nabla_{\theta^a} E[R_0^1(\tau)] \cdot \delta.
\]

For the LOLA update, we derive the following estimator for the second-order term in (4.2) based on policy gradients (see Supplementary Material for detailed derivation):

\[
\nabla_{\theta^1} \nabla_{\theta^2} E[R_a^0(\tau)]
\]

\[
= \mathbb{E} \left[ R_0^1(\tau) \nabla_{\theta^1} \log \pi^1(\tau) \nabla_{\theta^2} \log \pi^2(\tau) \right]
\]

\[
= \mathbb{E} \left[ \sum_{t=0}^{T} \gamma^t \gamma^t \left( \sum_{t=0}^{T} \nabla_{\theta^1} \log \pi^1(u_t^1|s_t) \right) \right].
\]

The complete LOLA update for agent 1 using policy gradients is

\[
f_{lola, pg}^1 = \nabla_{\theta^1} E[R_0^1(\tau)] \cdot \delta + \left( \nabla_{\theta^2} E[R_0^1(\tau)] \right)^T \nabla_{\theta^1} \nabla_{\theta^2} E[R_0^1(\tau)] \cdot \delta \eta.
\]

### 4.4 LOLA with Opponent Modeling

So far we have assumed that each agent has access to the exact parameters of the opponent. However, in adversarial settings these parameters are typically obscured and must be inferred from the state-action trajectories. Formally, we replace \( \theta^2 \) with \( \hat{\theta}^2 \), where \( \hat{\theta}^2 \) is estimated from trajectories using maximum likelihood:

\[
\hat{\theta}^2 = \arg\max_{\theta^2} \sum_t \log \pi_{\theta^2}(u_t^2|s_t)
\]

\( \hat{\theta}^2 \) then replaces \( \theta^2 \) in the LOLA update rule, both for the exact version using the value function and the gradient based approximation.
4.5 Higher Order LOLA

The LOLA learning rule so far assumes that the opponent is a naive learner that carries out first order learning using policy gradients. In this setting, which we call first-order LOLA, accounting for the learning of the other agent leads to a second order correction term. However, we can also consider a higher order LOLA agent that differentiates through the learning step of this first order LOLA agent, including the second order correction. This leads to a third order derivative in the correction term. While this third order term is typically difficult to compute using rollouts, when the exact value function is available it is tractable.

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). These two classical environments allow us to 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 exact discounted future reward can not be calculated and each player is parameterized with a deep policy network.

5.1 Iterated Games

The per-step payoff matrix of the prisoners’ dilemma is shown in Table 1.

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

Table 1: Payoff matrix of prisoners’ dilemma.

In a single-shot prisoners’ dilemma, there is only one Nash equilibrium by Fudenberg and Tirole (1991), where both agents defect. In the infinitely repeated prisoners’ dilemma, the folk theorem by Fudenberg and Tirole (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 0 for TFT and DD respectively.

IMP is a zero-sum game, with per-step payoffs 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.

|      | IPD      | IMP      |
|------|----------|----------|
| NL-Ex. | 20.8    | 0.0      |
| LOLA-Ex. | 81.0    | 98.8     |
| NL-PG  | 20.0    | 13.2     |
| LOLA-PG | 66.4    | 93.2     |

Table 3: 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.

We model the IPD and IMP as a two-agent MDP, where the state at time 0 is empty and at time t ≥ 1 is both agents’ actions from t − 1:

\[ s_t = (u_{t-1}^1, u_{t-1}^2) \quad \text{for } t > 1. \]

Each agent’s policy is fully parametrized by 5 probabilities. For agent a in the case of the IPD, they are \( \pi^a(C|s_0), \pi^a(C|CC), \pi^a(C|CD), \pi^a(C|DC) \) and \( \pi^a(C|DD) \). We can derive each agent’s future discounted reward as an analytical function of the agents’ policy (see Supplementary Material for details) and calculate the exact policy update for both NL and LOLA agents.

We further assume that agents can only update their policies between the rollouts, not during the iterated game play. Conceptually each agent submits their policy to the environment, which then get used to play a large number (batch size) of infinitely iterated games. Next both agents receive the traces resulting from these games and can submit updated policies to the environment.

5.2 Coin Game

Next we study LOLA in a setting that requires recurrent policies and features sequential actions. The ‘Coin Game’ was first proposed in Lerer and Peysakhovich (2017) as a higher dimensional expansion of the iterated prisoners’ dilemma with multi-step actions. As shown in Figure 5.2 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, each coin appearing when the last one has been picked up. Agents pick up coins by moving onto the field where the coin is located. While every agent receives a point for picking up a coin of any colour, whenever the ‘red agent’ picks up a blue coin the ‘blue agent’ loses 2 points and vice versa.

As a result, if both agents greedily pick up any coin available, they receive 0 points on average. In the ‘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 apply LOLA with opponent-modelling to this task.
Figure 3: In the coin game, two agents ‘red’ and ‘blue’, get rewarded for picking up coins. 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.

5.3 Training Details
In all our PG experiments we use gradient descent with step size 0.005 for the actor, 1 for the critic, and batch size 4000. 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.

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 LOLA agents behave in iterated games compared with NL agents?
2. Does replacing the exact policy update with policy gradient updates change the learned behaviors of LOLA and NL agents?
3. Does the learning of LOLA agents scale to high-dimensional settings where the agents’ policies are parametrized by deep networks?
4. When replacing access to the exact parameters of the opponent agent with opponent modeling, does LOLA agents’ behavior preserve?
5. 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
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 LOLA and NL describe pairs of agents. We consider mixed learning of one LOLA agent vs an NL agent in Section 6.3. Under NL, 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, in most cases the agents learn TFT. In particular agent 1 cooperates in s0, CC and DC, while agent 2 cooperates in s0, CC and CD. As a result, Figure 3c) shows that the average return per step is close to −1 for LOLA, corresponding to TFT, while NL results in an average reward of −2, corresponding to the fully defective (DD) equilibrium. Figure 3d) shows the average return per step for NL-PG and LOLA-PG where agents learn via policy gradient. LOLA-PG also demonstrates cooperation while agents defect in NL-PG. Further plots are provided in the Supplementary Material.

We conduct the same analysis for IMP. 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 IPD, LOLA agents learn policies consistent with TFT with a much higher probability and achieve higher reward 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 average returns per step for NL agents compared to the low variance under LOLA (0.37 vs 0.02).

6.2 Coin Game
As shown in Figure 4, NL agents collect coins indiscriminately, corresponding to defection. In contrast, LOLA agents learn to pick up coins predominantly (around 80%) of their own color, corresponding to cooperation. The same result holds true when agents have to learn the policy of the opponent, using LOLA with opponent modelling. We emphasize that in this setting 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.
6.3 Higher Order LOLA

In the exact value function setting the higher order LOLA terms can be evaluated. We use this to address the question of whether there is an arms race leading to ever higher orders of correction terms between the two agents. Table 4 shows that in IPD, a LOLA learner can achieve higher payouts against a naive learner. Thus, there is an incentive for either agent to switch from naive learning to first order LOLA. Furthermore, two LOLA agents playing against each other both receive higher rewards than a LOLA agent playing against a naive learner. This makes LOLA a dominant learning rule in IPD compared to naive learning. However, we further find that higher order LOLA provides no incremental gains when playing against a first order LOLA 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 / LOLA is a dominant learning rule in the space of all possible gradient-based rules, these initial results are encouraging.

|          | NL       | 1st order | 2nd Order |
|----------|----------|-----------|-----------|
| NL       | (-1.99, -1.99) | (-1.54, -1.28) | -         |
| 1st      | (-1.28, -1.54) | (-1.04, -1.04) | (-1.14, -1.17) |

Table 4: Higher order LOLA results on the IPD. A LOLA agent obtains higher rewards compared to a NL. 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 under 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 have shown that when both agents apply the LOLA learning rule, this leads to the emergence of cooperation based on tit-for-tat in the infinitely repeated iterated prisoners’ dilemma while independent learning does not. Empirical results show that in the IPD both agents are incentivized to use LOLA, while higher order exploits show no further gain. We also find that LOLA leads to stable learning of the Nash equilibrium in iterated matching pennies.

Furthermore we apply a policy gradient based version of LOLA to the ‘Coin Game’, a multi-step game which requires recurrent policies. In this setting, LOLA agents learn to collaborate, even when they do not have access to the policy of the other agent.

In the future we would like to address the exploitability of LOLA, when adversarial agents explicitly aim to take advantage of a LOLA learner using global search methods rather than gradient based methods. Just as LOLA is a way to exploit a naive learner, in principle there should be means of exploiting LOLA learners in turn, unless LOLA is itself an equilibrium learning strategy. We would also like to prove properties regarding the kind of equilibria to which LOLA agents converge. An initial step is to understand the dynamics in infinitesimal gradient ascent, similar to what was done by Wunder, Littman, and Babes [2010]. Another challenge is to apply LOLA in settings with many agents. Since each agent has to account for the learning of all other agents, total computational requirements are quadratic in the number of agents. One solution is to apply LOLA to a learned subset of opponents that have the greatest influence on the reward.
Acknowledgements

We would like to thank Ilya Sutskever, Bob McGrew, Paul Cristiano and the rest of OpenAI for fruitful discussion. We would like to thank Michael Littman for providing feedback on an early version of the manuscript.

References

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

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.

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.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017a. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926.

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

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

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

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.

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.

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.

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

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.; 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.

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

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

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

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

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 $\tau$. Then,

$$\nabla_{\theta^1} \nabla_{\theta^2} \mathbb{E}_{\pi_1, \pi_2, \tau} R_0^1(\tau) = \nabla_{\theta^1} \nabla_{\theta^2} \mathbb{E}_{\tau} \left[ R_0^1(\tau) \cdot \prod_{l=0}^{T} \pi^1(u_l^1|s_l, \theta^1) \cdot \prod_{l=0}^{T} \pi^2(u_l^2|s_l, \theta^2) \right]$$

$$\quad = \mathbb{E}_{\tau} \left[ R_0^1(\tau) \cdot \left( \nabla_{\theta^1} \left( \prod_{l=0}^{T} \pi^1(u_l^1|s_l, \theta^1) \right) \right) \left( \nabla_{\theta^2} \left( \prod_{l=0}^{T} \pi^2(u_l^2|s_l, \theta^2) \right) \right)^T \right]$$

$$\quad = \mathbb{E}_{\pi_1, \pi_2, \tau} \left[ R_0^1(\tau) \cdot \left( \nabla_{\theta^1} \left( \prod_{l=0}^{T} \pi^1(u_l^1|s_l, \theta^1) \right) \right) \left( \nabla_{\theta^2} \left( \prod_{l=0}^{T} \pi^2(u_l^2|s_l, \theta^2) \right) \right)^T \right]$$

The second equality is due to $\pi_1$ is only a function of $\theta_1$. The third equality is multiply and divide the probability of the episode $\tau$. The fourth equality factors the probability of the episode $\tau$ 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 $t$

$$\nabla_{\theta^1} \nabla_{\theta^2} \mathbb{E}_{\pi_1, \pi_2, \tau} r_t^1 = \mathbb{E}_{\pi_1, \pi_2, \tau} \left[ \left( \sum_{l=0}^{t} \nabla_{\theta^1} \log \pi^1(u_l^1|s_l, \theta^1) \right) \left( \sum_{l=0}^{t} \nabla_{\theta^2} \log \pi^2(u_l^2|s_l, \theta^2) \right)^T \right].$$

Sum the rewards over $t$,

$$\nabla_{\theta^1} \nabla_{\theta^2} \mathbb{E}_{\pi_1, \pi_2, \tau} R_0^1(\tau) = \mathbb{E}_{\pi_1, \pi_2, \tau} \left[ \sum_{t=0}^{T} \gamma^t r_t^1 \cdot \left( \sum_{l=0}^{t} \nabla_{\theta^1} \log \pi^1(u_l^1|s_l, \theta^1) \right) \left( \sum_{l=0}^{t} \nabla_{\theta^2} \log \pi^2(u_l^2|s_l, \theta^2) \right)^T \right],$$

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 $2 \times 2$ states depending on the last action taken.

As a consequence the policy of each agent can be represented by 5 parameters, $\theta^a$, 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
We denote $\theta^a = (\theta^{a,0}, \theta^{a,1}, \theta^{a,2}, \theta^{a,3}, \theta^{a,4})$. In these games the union of $\pi^1$ and $\pi^2$ induces a state transition function $P(s'|s) = P(u|s)$. Denote the distribution of $s_0$ as $p_0$:

$$p_0 = \begin{pmatrix}
\theta^{1,0} \theta^{2,0}, & \theta^{1,0} (1 - \theta^{2,0}), & (1 - \theta^{1,0}) \theta^{2,0}, & (1 - \theta^{1,0})(1 - \theta^{2,0})
\end{pmatrix}^T,$$

the payout vector as $r^1 = (-1, -3, 0, -2)^T$ and $r^2 = (-1, 0, -3, -2)^T$, and the transition matrix is

$$P = \begin{bmatrix}
\theta^1 \theta^2, & \theta^1 (1 - \theta^2), & (\theta^1 - 1) \theta^2, & (1 - \theta^1)(1 - \theta^2)
\end{bmatrix}$$

Then $V_1, V_2$ can be represented as

$$V^1(\theta^1, \theta^2) = p_0^T (r^1 + \sum_{t=1}^{\infty} \gamma^t P^t r^1)$$

$$V^2(\theta^1, \theta^2) = p_0^T (r^2 + \sum_{t=1}^{\infty} \gamma^t P^t r^2).$$

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

$$V^1(\theta^1, \theta^2) = p_0^T \frac{I}{I - \gamma P} r^1,$$

$$V^2(\theta^1, \theta^2) = p_0^T \frac{I}{I - \gamma P} r^2,$$

where $I$ is the identity matrix.

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

```
Figure 5: 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).
Figure 6: 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.