Learning to Cooperate

Akshat Agarwal
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
akshata@andrew.cmu.edu

Abstract

This project investigates the emergence of cooperative behavior between artificial agents in social dilemmas. The approach used, Learning with Opponent Learning Awareness (LOLA) is implemented from scratch, along with a novel gradient update rule formulation and opponent modeling through Expectation Maximization, which lead to improved learning with fewer assumptions.

1 Introduction

With the recent advances in artificial intelligence, multi-agent settings are becoming a highly relevant topic of research, considering that real world deployment of artificial agents will require them to interact with each other, in real time, and in meaningful ways. Multi-agent environments present a multitude of challenges not present in single-agent settings, most importantly the non-stationary induced in the environment from the point of view of any one agent. This is compounded by other problems like an exponentially growing state space, inevitable partial observability and even sparser rewards during learning. In 2015, Mnih et al. [5] started deep reinforcement learning, by combining convolutional nets with Q-learning. That has led to a spate of recent work in multi-agent reinforcement learning, a lot of it focusing on fully cooperative settings [3] and emergent communication [6]. These works often focus on giving a common payoff to the agents to strongly encourage collaboration. However, they overlook a strong element of social interaction, which is the emergence of reciprocity through self-interested behavior on the part of each agent. Both humans and animals (who hunt in packs in the wild) often realize that cooperating with others would be beneficial to them personally, and this is a behavior which needs to be intrinsic to artificial agents as well, for them to coexist with humans and integrate into our society. Therefore, getting self-interested agents to learn to cooperate is a worthwhile task.

In game theory and economics, such settings are modeled as social dilemmas, which are situations in which an individual profits from selfish behavior, unless everyone chooses the selfish alternative, in which case they all lose [1]. The Prisoners' Dilemma [8] is a popular testbed for research in social dilemmas. The payoff matrix for this game is given in Table 1 in which the Nash equilibrium rational outcome is for both players to Defect, even though the Pareto-optimal outcome would be for both to Cooperate. In the infinitely iterated setting of this game (IPD), agents can actually remember previous actions of their opponent, learn its strategy and change their own accordingly. This introduces an interesting element of learning to cooperate while preventing exploitative behavior by the other agent, and while there are infinitely many Nash equilibria for these games (due to the Folk theorem [9]), Tit-For-Tat (TFT) is one of the best performing strategy here [2]. TFT basically involves starting out with cooperation, and then repeating the previous action of the opponent. Strategies like always defecting would be suboptimal (with a payoff of -2), especially given that there is an opportunity for the agents to learn to cooperate and get a better payoff of -1. This is a simple game which incorporates the essence of the problem of getting self-interested agents to cooperate, and has hence been used as the testbed for evaluating algorithms.

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

Table 1: Payoff matrix for the Prisoners' Dilemma. C: Cooperate, D: Defect

While most multi-agent learning algorithms learn to defect with high probability in IPD, Learning with Opponent Learning Awareness (LOLA) [4] is a novel approach which reasons about and anticipates the learning of other agents, and accounts for that while updating its own strategy. LOLA learners were able to learn the TFT strategy in the IPD, indicating emergent reciprocity. LOLA learners can cooperate with other LOLA learners, exploit naive learners and prevent exploitation by higher-order LOLA learners, according to the results of the paper. Since this work presents a unique and novel approach, in this project I do a deep dive into the algorithm presented, understanding the equations presented and using them to formulate gradient update rules. I also study Higher-Order LOLA, with the objective of understanding why it does not beat the first-order version. Apart from this, I also successfully remove a major assumption made in the paper, that of being able to fully observe the opponent's strategy, by implementing an expectation-maximization algorithm which requires each agent to observe only the opponent's actions, use that to create a model of its strategy, and use that belief to make its own updates. This version, with opponent modeling, works just as well as the original in terms of final strategy learnt, however is computationally more expensive due to the added cost of modeling strategy, and using separate models for each agent in their updates.

2 Problem Formulations

Let $V^a(\theta^1, \theta^2)$ be the expected total discounted return for agent a, π^a be its policy which is parameterized by θ^a, where $a = 1, 2$.

2.1 Naive Learner

A naive learner optimizes its strategy to be a best response to the opponent's current strategy. Mathematically, this can be expressed as:

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

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

Since it is difficult to optimize over the entire parameter space, these equations can be straightforwardly approximated by a gradient update rule:

$$\theta^1_{i+1} = \theta^1_{i+1} + \frac{\partial V^1(\theta^1_i, \theta^2_i)}{\partial \theta^1_i} \cdot \delta \tag{3}$$

and similarly for the other agent.

2.2 LOLA Learner

A LOLA Learner optimizes its own policy by accounting for the fact that the opponent is learning as well. So, instead of optimizing against the opponent's current strategy, it optimizes its strategy for the opponent's best response to its changed strategy, hence maximizing its expected discounted future return. Mathematically, this can be expressed as:

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

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

Again, optimizing over the entire policy space is a global optimization problem which is difficult for the agents to do considering that they would not have access to the expected discounted return (or its

gradients) at all parameter values. To accommodate this, a local gradient-based rule is formulated by writing the gradient of V^1 with respect to θ^1, doing a Taylor expansion of that and then approximating by ignoring higher order terms, as given below.

The optimization objective is

$$\theta^1 = \arg\max_{\theta^1} V^1(\theta^1, \arg\max_{\theta} V^2(\theta^1, \theta))$$

The second argument is a Naive Learner, so by Eqn. 3 we get

$$\theta^1 = \arg\max_{\theta^1} V^1(\theta^1, \theta^2 + \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \cdot \eta)$$

Considering the second argument to be a single value, we can again use the Naive Learner argument to write this as:

$$\theta^1 = \theta^1 + \frac{\partial}{\partial \theta^1} V^1(\theta^1, \theta^2 + \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \cdot \eta) \cdot \delta$$

Now, Taylor's theorem to a first order approximation is:

$$f(x, y + \Delta y) = f(x, y) + \frac{\partial f}{\partial y} \Delta y$$

Which gives us

$$V^1(\theta^1, \theta^2 + \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \cdot \eta) = V^1(\theta^1, \theta^2) + \frac{\partial V^1(\theta^1, \theta^2)}{\partial \theta^2} \cdot \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \cdot \eta$$

Substituting this in above obtained equations:

$$\theta^1 = \theta^1 + \frac{\partial}{\partial \theta^1} \left(V^1(\theta^1, \theta^2) + \frac{\partial V^1(\theta^1, \theta^2)}{\partial \theta^2} \cdot \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \cdot \eta \right) \cdot \delta$$

which gives

$$\theta^1 = \theta^1 + \delta \cdot \frac{\partial}{\partial \theta^1} V^1(\theta^1, \theta^2) + \delta \cdot \eta \left(\frac{\partial^2 V^2(\theta^1, \theta^2)}{\partial \theta^1 \partial \theta^2} \cdot \frac{\partial V^1(\theta^1, \theta^2)}{\partial \theta^2} + \frac{\partial^2 V^1(\theta^1, \theta^2)}{\partial \theta^1 \partial \theta^2} \cdot \frac{\partial V^2(\theta^1, \theta^2)}{\partial \theta^2} \right) \tag{4}$$

This is the first order LOLA update equation. It has second order differential terms, which essentially let the agent reason about 2 steps, its own as well as the agent's (as opposed to the naive learner where the agent reasons only about its own next step). Note that Agent 2's update equation will be derived very similarly, and has been omitted for brevity.

This is a different gradient formulation than that presented in the paper, and gets stronger exploitation results against naive learners.

2.3 Higher-Order LOLA

Despite investing a lot of effort to get higher order LOLA agents working, it did not happen. However, I would like to report my formulations of the higher-order problem. Writing it in the $\arg\max$ form is tricky because of recursiveness, but in the differential gradient form it can be written as:

$$\theta^1 = \theta^1 + \frac{\partial}{\partial \theta^1} V^1(\theta^1, \theta^2 + \frac{\partial}{\partial \theta^2} V^2(\theta^1 + \frac{\partial V^1(\theta^1, \theta^2)}{\partial \theta^1} \cdot \psi, \theta^2) \cdot \eta) \cdot \delta$$

This can be simplified to:

$$\theta^1 = \theta^1 + \delta \cdot \frac{\partial}{\partial \theta^1} V^1 + \delta \cdot \eta \left(\frac{\partial^2 V^2}{\partial \theta^1 \partial \theta^2} \cdot \frac{\partial V^1}{\partial \theta^2} + \frac{\partial^2 V^1}{\partial \theta^1 \partial \theta^2} \cdot \frac{\partial V^2}{\partial \theta^2} \right) +$$

$$\delta \cdot \eta \cdot \psi \left(\frac{\partial V^1}{\partial \theta^2} \frac{\partial V^2}{\partial \theta^1} \frac{\partial^3 V^1}{\partial \theta^1 \partial \theta^2 \partial \theta^1} + \frac{\partial^2 V^1}{\partial \theta^2 \partial \theta^1} \frac{\partial V^1}{\partial \theta^2} \frac{\partial^2 V^2}{\partial \theta^1 \partial \theta^1} + \frac{\partial^2 V^1}{\partial \theta^2 \partial \theta^1} \frac{\partial V^2}{\partial \theta^1} \frac{\partial^2 V^1}{\partial \theta^1 \partial \theta^2} \right)$$

3 Experiments

As mentioned above, the Iterated Prisoners' Dilemma is used as a testbed. The game is modeled as a two-agent MDP, where the state at time 0 is empty (Φ) and at any time t is both agents' actions from $t-1$:

$$s_t = (a^1_{t-1}, a^2_{t-1}) \triangleq \mathbf{a} \tag{5}$$

Note that there are 5 unique states an agent could be in at any point: {Φ,CC,CD,DC,DD}. Since there are only 2 actions, Cooperate (C) and Defect (D), whose probabilities have to sum up to 1, each agent's policy can be represented by only 5 parameters:

$$\theta^a = \begin{bmatrix} \pi^a(C|\Phi) \\ \pi^a(C|CC) \\ \pi^a(C|CD) \\ \pi^a(C|DC) \\ \pi^a(C|DD) \end{bmatrix} \tag{6}$$

The reward functions can also be expressed as vectors, with $r^1 = [-1, -3, 0, -2]^T$ and $r^2 = [-1, 0, -3, -2]^T$. Also, since the game can be easily modeled, we can obtain the exact future discounted reward as a function of the policy (as given in the Appendix in [4]), getting:

$$V^a(\theta^1, \theta^2) = p_0^T (I - \gamma P)^{-1} r^a \tag{7}$$

where P is the state transition matrix $P(s'|s) = P(\mathbf{a}|s)$, and γ is the discount factor, which encodes how much the agent cares about future rewards compared to immediate reward. $\gamma = 1$ means the agent cares equally about all future rewards, no matter how far in the future, while $\gamma = 0$ means the agent does not care at all about future rewards.

During training, δ was set to 0.1, while η and γ were varied to understand their effect on the learning. Gradients and Hessians were calculated using PyTorch [7].

3.1 Opponent Modeling

As can be seen in Eqn. 4, the update for agent 1 requires access to the other agent's policy parameters. This is not a very realistic assumption in adversarial settings, or even cooperative settings with communication bandwidth constraints. So, I implemented a small Expectation-Maximization step in every update, where the hidden (latent) variable for each agent is the other agent's policy. Observing only the actions of other agents, and with full knowledge of the state, each agent forms a belief of the policy parameters of the other agent (the **E-step**), and then use that belief in their value function estimation (Eqn. 7) as well as parameter update (Eqn. 4) to maximize their expected total discounted reward (the **M-step**).

$$\hat{\theta^2} = \arg\max_{\theta} \sum_t \log \pi_\theta(\mathbf{a}^2_t | s_t) \tag{8}$$

where $\hat{\theta^2}$ is the estimate of agent 2's policy formed by agent 1. This fully decentralizes the LOLA learning rule and removes assumption of any communication between the two agents.

3.2 Results

Results are given in Table 2. π^1 and π^2 are the learned policy parameters for agent 1 and 2, respectively. The 5 numbers correspond to the 5 parameters in Eqn. 6. Average reward R1 and R2 was measured over 100 rollouts with 10000 steps each. Some key observations here are:

- Two NL agents learn to defect with very high probability. This strategy gets an average reward of -2.
- Two LOLA agents learn the Tit-For-Tat strategy, which means they start with cooperation, and then repeat the previous action of the opponent. This strategy gets an average reward of -1.
- As the time horizon is shortened (by reducing γ), the LOLA learners' performance degrades. This is expected since having a shorter horizon means the game resembles a finitely iterated PD more closely, in which the agents learn to defect at the last turn.

- LOLA is able to exploit the NL agent to get a reward higher than it got against a LOLA agent. The payoff my gradient update rule gets (in bold) here is higher than reported in the paper, which was a payoff of -1.54 for NL agent and -1.28 for LOLA agent (corresponding to less exploitation).

Agent1	Agent 2	η	γ	R1	R2	π^1	π^2	Strategy
NL	NL	3	0.96	-2	-2	0,0.27,0,0,0	0,0.27,0,0,0	Defect
LOLA	LOLA	3	0.96	-1	-1	1,1,0,1,0	1,1,1,0,0	TFT
LOLA	LOLA	10	0.8	-1.05	-1.05	1,1,0,1,0	1,1,1,0,0	TFT
LOLA	LOLA	20	0.6	-2	-2	0,0.5,0.1,0.45,0	0,0.5,0.45,0.1,0	–
NL	LOLA	3	0.96	**-1.96**	**-0.62**	0.66,1,1,1,0	1,0.5,0.5,0,0	–
NL	LOLA	10	0.8	-1.06	-0.97	0.85,1,1,0.57,0.1	0,1,0.26,0,0	–

Table 2: Results

4 Conclusion

I implemented, and released[1] the first publicly available implementation of the LOLA algorithm. I formulated my own gradient update rule using a Taylor expansion of the LOLA fundamental equation, which performed better than the update rule used in the original paper. I also implemented opponent modeling to remove the requirement of access to opponent's policy parameters, which did not change the final policy learnt (though it did take more steps to converge, and each step was computationally more expensive).

Acknowledgments

I'd like to thank Prof. Michael Erdmann for being an awesome professor with a really unique and passionate style of teaching, and giving students the space to experiment on tangential research while still getting credited for it, through this project.

References

[1] Scott T Allison, James K Beggan, and Elizabeth H Midgley. The quest for" similar instances" and" simultaneous possibilities": Metaphors in social dilemma research. *Journal of Personality and Social Psychology*, 71(3):479, 1996.

[2] R Axelrod. The evolution of cooperation, 1984.

[3] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

[4] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[6] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[8] William Poundstone. Prisoner's dilemma: John von neuman, game theory, and the puzzle of the bomb, 1992.

[1] `https://github.com/agakshat/LOLA-pytorch`

[9] B Myerson Roger. Game theory: analysis of conflict, 1991.