

Game-Theoretic Multi-Agent Reinforcement Learning



NIKOLAS KIRSCHSTEIN

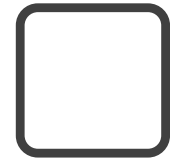
A major project essay for
COMP4121 Advanced Algorithms

1 December 2024

Abstract

Future AI systems will be deployed in highly dynamic and complex environments such as road traffic, search-and-rescue operations, smart power grids, stock markets, and the Internet of Things. An effective and ethical AI agent must be able to 1) handle a temporally evolving environment and 2) take into account the presence of other intelligent agents. In this essay, I elucidate how issue 1) is addressed by reinforcement learning (RL) theory and 2) is addressed by game theory, to then fuse the two to arrive at multi-agent reinforcement learning algorithms. In particular, I explain a sophisticated technique called Learning with Opponent-Learning Awareness (LOLA), which makes an agent aware of the learning process of other agents rather than treating them as a static part of the environment, and demonstrate that it achieves social welfare in simple, partially competitive settings. My work serves as a rapid introduction to the relevant concepts for aspiring researchers digging into multi-agent RL, as well as a showcase for how small but clever algorithmic tweaks can greatly impact the emergent behaviour of AI systems in the non-cooperative case.

Contents



1	Introduction	1
2	Game Theory Foundations	3
2.1	One-Shot Games	3
2.2	Repeated Games	6
3	Single-agent Reinforcement Learning	8
3.1	The Framework	8
3.2	Policy Gradient Algorithms	9
4	Multi-agent Reinforcement Learning	12
5	Learning with Opponent-Learning Awareness	15
5.1	The Algorithm	15
5.2	Emergent Behaviour	17
6	Conclusion	21
	Bibliography	22

Notation



x	A scalar
$\boldsymbol{x}$	A vector
x_i	Element i of $\boldsymbol{x}$
$\boldsymbol{x}_{-i}$	Subvector of all elements except i in $\boldsymbol{x}$

Warning: I use numerator layout* for derivatives

$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$	Gradient of scalar function f w.r.t. $\boldsymbol{x}$ as a <i>row</i> vector
$\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$	Jacobian of vector function $\boldsymbol{f}$ w.r.t $\boldsymbol{x}$ with gradients in the <i>rows</i>
$\nabla_{\boldsymbol{x}} \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$	Hessian of scalar function f w.r.t $\boldsymbol{y}$ then $\boldsymbol{x}$ with shape $ \boldsymbol{y} \times \boldsymbol{x} $

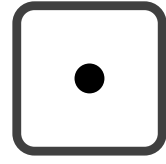
$p(\boldsymbol{x})$	A joint distribution of the random variable $\boldsymbol{x}$
$\mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}$	Expected value w.r.t. $\boldsymbol{x}$ drawn from p

X	A set
$\mathbb{1}_X$	The indicator function of the set X
X^Y	The set of functions $Y \rightarrow X$ (in particular, $X^{\mathbb{N}}$ are the sequences in X)
$\Delta(X)$	Probability simplex over the finite set X (i.e., $\{p : X \rightarrow [0, 1] \mid \sum_{x \in X} p(x) = 1\}$)

*see https://en.wikipedia.org/wiki/Matrix_calculus#Layout_conventions

Chapter 1

Introduction



Multi-agent *reinforcement learning* (RL) is a theoretical framework that models autonomous entities, called *agents*, who have individual goals and decision-making capabilities and are influenced by each other's decisions. Multi-agent RL has shown remarkable success in complex environments, with recent examples including AlphaGo beating the European champion in the board game Go (Silver et al., 2016) and AlphaStar reaching Grandmaster level in the strategy game StarCraft II (Vinyals et al., 2019).

While the zero-sum game Go is purely competitive, the general-sum game StarCraft II exhibits elements of both competition and cooperation. This is a crucial aspect as future applications like fully autonomous cars will, next to coordination, entail competition and conflict. Algorithmically equipping AI agents to be able to cooperate when it makes sense while still being competitive is hence a fundamental problem that generalises well beyond these game settings to virtually any context of multiple intelligent agents that interact strategically. This strategic interaction, in turn, is the object of study in *game theory*, which, contrary to the name*, applies to a broad range of problems, especially in economics and informatics.

The goal of this essay is twofold. First, I provide an introduction to multi-agent RL from first principles while taking on a game-theoretic lens. I demonstrate the close relationships between game theory and reinforcement learning to highlight common ideas and enhance the understanding of multi-agent RL. My carefully chosen notation and structure will make these connections very apparent. Second, I discuss multi-agent RL algorithms and their emergent behaviour in infinitely repeated games of Matching Pennies and the Prisoner's Dilemma, which are fundamental testbeds in game theory. In particular, I elucidate how a particular multi-agent RL method, called Learning with Opponent-Learning Awareness (LOLA) can achieve cooperation to emerge in partially competitive settings. Beforehand, I shall provide a brief literature review.

*It is indeed a misnomer. Calling it *strategic interaction theory* would do it more justice.

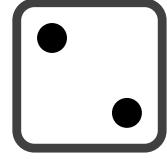
As mentioned, game theory considers interactions of rational agents with each other who seek to maximise their own utility. Mathematicians have been studying strategic behaviour in very specific settings for a long time. One example is the first formal treatment of chess strategies by [Zermelo \(1913\)](#). Game theory as a generic discipline, however, was ignited by [Neumann & Morgenstern \(1944\)](#). [Nash \(1951\)](#) then introduced the seminal concept of a Nash equilibrium. The iconic Prisoner’s Dilemma was conceived by [Flood \(1952\)](#). A plethora of more recent work in game theory exists but is out-of-scope for the argumentation I intend in this essay.

Reinforcement learning considers temporal interaction of a single agent with an environment using a reward-based system. The idea dates back to the observation of animals discovering new beneficial behaviours via trial-and-error and then reinforcing them ([Thorndike, 1898](#)). Based on this, [Minsky \(1954\)](#) invented neural-analog reinforcement calculators. At the same time, the field of dynamic programming ([Bellman, 1954](#)) was born. This field evolved into modern-day reinforcement learning through significant contributions by [Sutton & Barto \(1998\)](#). The foundation for modern-day state-of-the-art policy gradient methods was laid by [Williams \(1992\)](#) and [Sutton et al. \(1999\)](#).

Merging game theory and RL results in the study of multi-agent RL. A standard and straightforward approach is independent RL ([Matignon et al., 2012](#)) that simply ignores the interaction between agents. While it works well in cooperative multi-agent settings, partially competitive settings require an algorithmic treatment of agent-agent interaction. One prominent such approach is LOLA ([Foerster et al., 2018](#)), which explicitly reasons about the learning behaviour of other agents.

The remainder of this essay is structured as follows: Chapter 2 explains the fundamental concepts of game theory that we require moving forward, and separately from that Chapter 3 introduces RL as a sequential decision-making process. Chapter 4 then performs the marriage of game theory and RL into multi-agent RL, carving out similarities and differences. Chapter 5 explores multi-agent algorithms in the context of non-cooperative settings, with a comprehensive treatment of the state-of-the-art LOLA algorithm. Chapter 6 concludes the essay.

Game Theory Foundations



The theoretical foundation for strategic interaction among rational decision-makers is provided by game theory. Section 2.1 describes the notion of a normal-form game, analyses two example games that I will continually refer back to throughout the essay, and establishes the solution concept of a Nash equilibrium. Section 2.2 considers indefinitely repeated normal-form games, with particular focus on the Prisoner's Dilemma.

2.1 One-Shot Games

Definition 1 (Normal-Form Game). A *normal-form game* is a 3-tuple $(N, (A_i)_{i \in N}, (u_i)_{i \in N})$, consisting of

- a finite set of *players* $N = \{1, \dots, n\}$,
- a finite set of *actions* A_i per player $i \in N$, and
- a *utility* function $u_i : A_1 \times \dots \times A_n \rightarrow \mathbb{R}$ per player $i \in N$.

Definition 2 (Action Profile). The set of *action profiles* is $A := A_1 \times \dots \times A_n$.

Normal-form games are fully specified by the lookup tables of the utility functions and can thus be represented by their *payoff matrix*, a tensor of shape $|A_1| \times \dots \times |A_n| \times n$ whose entry $(a_1, \dots, a_n, i)$ corresponds to $u_i(a_1, \dots, a_n)$. We will see this in Examples 1 and 2 shortly. Typically, the following broad classification is made:

- **common-interest games**: all players share the same utility function:

$$u_1 = \dots = u_n$$

- **zero-sum games**: the player's utilities are diametrically opposed:

$$\sum_{i=1}^n u_i(\mathbf{a}) = 0 \quad \text{for all } \mathbf{a} \in A$$

- **general-sum games**: neither common-interest nor zero-sum

Since in common-interest games, all players are symmetrical, the problem shifts from individual strategising to payoff distribution among the players. This is the subject of cooperative game theory, which we do not consider here. Instead, we focus on games that contain elements of competition, i.e., zero-sum and general-sum games.

Example 1 (Matching Pennies). In this two-player zero-sum game, which can be regarded as a simplified version of rock-paper-scissors, each player secretly puts their penny heads-up or tails-up to then compare their pennies. The first player wins if the pennies match while the second player wins if they mismatch:

	heads	tails
heads	$(+1, -1)$	$(-1, +1)$
tails	$(-1, +1)$	$(+1, -1)$

Table 1: Payoff matrix of Matching Pennies.

However, most of life actually is not a zero-sum game where one player increasing their payoff necessarily means decreasing the other players' payoffs. There are win-win situations, and lose-lose situations.

Example 2 (Prisoner's Dilemma; Flood (1952)). In this famous two-player game general-sum game, each player is a suspect during an individual police interrogation, with two actions "cooperate" (i.e., remain silent) and "defect" (i.e., rat on the other player). If one player defects and the other cooperates, the defector is set free immediately and the cooperator receives the full sentence. If both defect, they share the sentence duration. However, if both players cooperate, they are only put in short-term custody. We encode this as follows:

	cooperate	defect
cooperate	$(-1, -1)$	$(-3, 0)$
defect	$(0, -3)$	$(-2, -2)$

Table 2: Payoff Matrix of Prisoner's Dilemma.

To see why the Prisoner's Dilemma is indeed a dilemma, we introduce a machinery of formal terminology that is quintessential to game theory.

Definition 3 (Strategy). The set of (*mixed*) *strategies* of player i in a normal-form game are the Multinoulli distributions $\Delta(A_i)$ over A_i . Recall that

$$\Delta(A_i) = \{\pi : A_i \rightarrow [0, 1] \mid \sum_{a \in A_i} \pi(a) = 1\}.$$

Definition 4 (Strategy Profile). A *strategy profile* is a tuple of strategies for every player, i.e., an element $\boldsymbol{\pi}$ of $\Delta(A_1) \times \cdots \times \Delta(A_n)$.

Definition 5 (Expected Utility). The *expected utility* of a strategy profile $\boldsymbol{\pi}$ from the perspective of player i is

$$u_i(\boldsymbol{\pi}) := \sum_{\mathbf{a} \in A} \left(\underbrace{\prod_{i=1}^n \pi_i(a_i)}_{\text{probability of } \mathbf{a} \text{ being played}} \cdot u_i(\mathbf{a}) \right)$$

Definition 6 (Best Response). The set of *best responses* of player i to a profile π_{-i} of opponent strategies is

$$\text{BR}(\pi_{-i}) := \arg \max_{\pi \in \Delta(A_i)} u_i(\pi, \pi_{-i}).$$

Remark. The subscript $\cdot_{-i}$ is syntactic sugar to refer to the $n - 1$ opponents of agent i .

Game theory offers *solution concepts* which are principled recommendations of what strategies to employ. The most widely used and well-understood solution concept is based on the notion of best responses.

Definition 7 (Nash Equilibrium). A *Nash equilibrium* is a strategy profile π consisting of mutual best responses, i.e., $\pi_i \in \text{BR}(\pi_{-i})$ for all $i \in N$.

The corresponding existence theorem (Nash, 1951) guarantees that at least one Nash equilibrium always exists in finite games. While computing one such equilibrium is PPAD-complete in general (Daskalakis et al., 2006), we can calculate them exactly for our two example games.

Lemma 1. *Matching Pennies (Example 1) has a unique Nash equilibrium consisting in a 50/50 strategy for both players.*

Proof. Let π_1, π_2 be the strategies of player 1 and 2, respectively, where they play heads with probabilities $p_1, p_2 \in [0, 1]$. Player 1's best response to π_2 is tails if $p_2 < 0.5$, heads if $p_2 > 0.5$, and any convex combination if $p_2 = 0.5$. Player 2's best response to π_1 is heads if $p_1 < 0.5$, tails if $p_1 > 0.5$, and any convex combination if $p_1 = 0.5$. Thus, the only possibility for both strategies to be best responses to each other is $p_1 = p_2 = 0.5$. $\square$

Lemma 2. *Prisoner's Dilemma (Example 2) has a unique Nash equilibrium consisting in both players defecting no matter what.*

Proof. For every action of the opponent, the only best response is to defect. Consequently, this is also the unique best response to any mixed strategy. $\square$

Lemma 2 implies that in the Prisoner's Dilemma, two rational players would always defect on each other (equilibrium strategy profile), resulting in both being imprisoned. If, however, both had cooperated, they would have been free after one week. Hence, the Prisoner's Dilemma is indeed a social dilemma. It applies to real-world problems as well, such as international relations: two countries are economically better off removing trade barriers, but if only one country does it unilaterally, it puts itself at a disadvantage. Analogous considerations hold for the example of a nuclear arms race.

We can conclude that, while Nash equilibrium is a solid solution concept for rational players trying to maximise their own utility, it does not necessarily lead to social welfare, as exemplified by the Prisoner's Dilemma.

2.2 Repeated Games

Most strategic interactions occurring in life are not limited to a single encounter like the one-shot games in the last section. A natural question is how the dynamics change when basic normal-form games such as Examples 1 and 2 are played repeatedly. Consider an infinite sequence of playing a normal-form game $(N, (A_i)_{i \in N}, (u_i)_{i \in N})$. At every time step t , player i follows some strategy $\pi_i^{(t)}(\mathbf{a}^{(0)}, \dots, \mathbf{a}^{(t-1)}) \in \Delta(A_i)$ conditioned on the past history of actions in the game. The *average payoff* of this player i is then given by the geometrically weighted average of the individual utilities:

$$\bar{u}_i := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t u_i(\pi_1^{(t)}, \dots, \pi_n^{(t)}).$$

We can interpret this as playing the base game indefinitely for a random, γ -geometrically distributed number of time steps. One crucial property of this is that, due to the memorylessness of the geometric distribution, no matter how many rounds have been played, the expected number of rounds left is constant, namely γ^{-1} .

The indefiniteness can significantly change the game dynamics: because no move is necessarily the last, a player must always consider the effect of their actions on the future behaviour and information of the opponents. For Repeated Matching Pennies (RMP), this is not the case, as the one-shot equilibrium strategy is a uniform randomisation and any deviation from it would be exploited by the opponent in subsequent rounds. In the Repeated Prisoner's Dilemma (RPD), there similarly exists the trivial sequential equilibrium consisting in always defecting, achieving an average payoff of -2 for both players.

Definition 8 (Always-Defect (DD) Strategy).

$$\pi_{i, \text{DD}}^{(t)}(\mathbf{a}^{(0)}, \dots, \mathbf{a}^{(t-1)}) := \text{defect}$$

However, and most notably, this is not the only sequential equilibrium but, in fact, there exist infinitely many (Myerson, 2013)! This means that cooperation becomes a viable strategy when the Prisoner's Dilemma is repeated indefinitely. Even more, Axelrod & Hamilton (1981) empirically observed that in RPD, strategies based on cooperation turned out most successful when run against a variety of strategies. They identified four key properties that characterised the most successful strategies:

1. *nice*: never be the first to defect
2. *provocable*: respond to uncalled-for defection, and do it soon
3. *forgiving*: return to cooperation after responding to a provocation
4. *clear*: stay predictable enough to enable other players to establish trust

Interestingly, the top strategy was observed to be the simplest possible materialisation of the above properties: starting with cooperation and then reciprocating the opponent's most recent action. This is a principle well-known as *tit-for-tat* and it achieves a better payoff of -1 for both players.

Definition 9 (Tit-For-Tat (TFT) Strategy).

$$\pi_{i,\text{TFT}}^{(t)}(\mathbf{a}^{(0)}, \dots, \mathbf{a}^{(t-1)}) := \begin{cases} \text{cooperate} & \text{if } t = 0 \\ a_{-i}^{(t-1)} & \text{if } t \geq 1 \end{cases}$$

As mentioned, the RPD has infinitely many sequential equilibria. Indeed, it is intuitive that TFT is one of them: If we know the opponent plays TFT, our maximum possible average payoff is $(1 - \gamma) \sum_{t=0}^{\infty} \gamma^t (-1)$, as any defection from our side will be retaliated immediately, leading to a worse payoff. The optimal payoff is achieved, among other strategies, by TFT itself, rendering it a best response. Another noteworthy cooperative sequential equilibrium is the one where both play cooperate until one defects, whereafter both defect forever. Yet, this is not a forgiving strategy.

Overall, we have seen that when a normal-form game is repeated, new spaces of possible strategies can open up. Sometimes, like in the RPD, these include new Nash equilibria that are strictly more socially beneficial than the ones in the base game. We will revisit the RPD, and also RMP, later in Chapter 5 when we discuss the emergent behaviour in multi-agent RL. Prior to that, we need to precisely define RL in the first place.

Single-agent Reinforcement Learning



Normal-form games only consider a time singularity with a single decision being taken, after which “the world ends”. However, the real world does not do that. Instead, we would like to model sequences of multiple decisions in an environment that is affected by these decisions. In the case of a single agent, this is what reinforcement learning achieves. Section 3.1 fully specifies the theoretical framework, based on which Section 3.2 outlines an important class of RL algorithms, policy gradient methods.

3.1 The Framework

The core modelling concept of RL is based on the well-known notion of a Markov chain which models sequential stochastic processes. It gets enriched by an element of decision-making, including a quantification of the consequences thereof, to account for the autonomy of agents.

Definition 10 (Markov Decision Process). A *Markov Decision Process (MDP)* is a 5-tuple (S, A, p, r, γ) , consisting of

- a finite set of *states* S ,
- a finite set of *actions* A ,
- a conditional *transition probability* distribution $p : S \times A \rightarrow \Delta(S)$,
- a *reward* function $r : S \times A \rightarrow \mathbb{R}$ which is bounded, and
- a *discount factor* $\gamma \in [0, 1)$.

An *episode* of the MDP is executed as follows: the agents starts in an initial state $s^{(0)} \in S$. At time t it chooses an action $a^{(t)}$ that results in an immediate reward $r(s^{(t)}, a^{(t)})$ and a new state $s^{(t+1)}$ with probability $p(s^{(t+1)} \mid s^{(t)}, a^{(t)})$. The way in which the agent chooses an action in a given state is governed by its *policy*.

Definition 11 (Policy). A *policy* is any function $\pi : S \rightarrow \Delta(A)$ that maps a state to a Multinoulli distribution over actions.

An MDP together with a policy define a complete probabilistic model. The fundamental object for this is a *trajectory* $\tau = ((s^{(t)}, a^{(t)}, r^{(t)}))_{t=0}^{\infty} \in (S \times A \times \mathbb{R})^{\mathbb{N}}$ consisting of all states, actions, and rewards obtained during the episode.

Proposition 1 (Probabilistic Model of RL). *Provided some policy π , the probability of a trajectory τ is given by*

$$p(\tau \mid \pi) = \prod_{t=0}^{\infty} \pi(a^{(t)} \mid s^{(t)}) \cdot \mathbb{1}_{\{r(s^{(t)}, a^{(t)})\}}(r^{(t)}) \cdot p(s^{(t+1)} \mid s^{(t)}, a^{(t)}). \quad (3.1)$$

Proof. The chain rule of probability together with the Markovianity of transitions, rewards, and policies yields

$$\begin{aligned} p(\tau \mid \pi) &= \prod_{t=0}^{\infty} p(a^{(t)}, r^{(t)}, s^{(t+1)} \mid s^{(t)}, r^{(t-1)}, a^{(t-1)}, s^{(t-1)}, \dots) \\ &= \prod_{t=0}^{\infty} p(s^{(t+1)} \mid \cancel{r^{(t)}}, a^{(t)}, s^{(t)}, \cancel{r^{(t-1)}}, \cancel{a^{(t-1)}}, \cancel{s^{(t-1)}}, \dots) \\ &\quad \cdot p(r^{(t)} \mid a^{(t)}, s^{(t)}, \cancel{r^{(t-1)}}, \cancel{a^{(t-1)}}, \cancel{s^{(t-1)}}, \dots) \\ &\quad \cdot p(a^{(t)} \mid s^{(t)}, \cancel{r^{(t-1)}}, \cancel{a^{(t-1)}}, \cancel{s^{(t-1)}}, \dots) \\ &= \prod_{t=0}^{\infty} p(s^{(t+1)} \mid a^{(t)}, s^{(t)}) \underbrace{p(r^{(t)} \mid a^{(t)}, s^{(t)})}_{\mathbb{1}_{\{r(s^{(t)}, a^{(t)})\}}(r^{(t)})} \underbrace{p(a^{(t)} \mid s^{(t)})}_{\pi(a^{(t)} \mid s^{(t)})}. \quad \square \end{aligned}$$

Based on this probabilistic model, the RL problem now consists in finding an optimal policy that maximises the expected discounted sum of rewards:

$$\pi^* \in \arg \max_{\pi} J(\pi) := \mathbb{E}_{\tau \sim p(\tau \mid \pi)} \left[\underbrace{\sum_{t=0}^{\infty} \gamma^t r^{(t)}}_{=: r(\tau)} \right]. \quad (3.2)$$

Geometric discounting via the factor γ is mainly a technical necessity to ensure convergence of the sum. Yet, it also allows for the semantic interpretation of prioritising rewards in the near future over the distant future. To avoid short-sightedness, the factor is commonly chosen very close to one, e.g., $\gamma = 0.99$.

3.2 Policy Gradient Algorithms

A widely used approach to maximising (3.2) consists in using a parametric family of functions π_{θ} (e.g., an artificial neural network) parametrised by $\theta \in \mathbb{R}^d$ and solving for the optimal parameters via stochastic gradient ascent on

$$J(\theta) := \mathbb{E}_{\tau \sim p(\tau \mid \pi_{\theta})} [r(\tau)]. \quad (3.3)$$

To approximate the gradient of (3.3), one would like to push the differentiation inside the expectation to enable Monte Carlo estimation, i.e., averaging over independently drawn

trajectories. However, this is not directly possible as the expectation itself depends on the differentiation variable $\boldsymbol{\theta}$. A reformulation afforded by the following theorem comes to the rescue.

Theorem 1 (Policy Gradient Theorem, [Sutton et al. \(1999\)](#)). *The gradient of (3.3) is Monte Carlo-estimable as the following equality holds:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p(\tau|\pi_{\boldsymbol{\theta}})}[r(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau | \pi_{\boldsymbol{\theta}})]. \quad (3.4)$$

Proof. The key trick is that $\frac{d}{dx} \log(x) = \frac{1}{x}$ and hence $\nabla_{\mathbf{x}} \log f(\mathbf{x}) = \frac{\nabla_{\mathbf{x}} f(\mathbf{x})}{f(\mathbf{x})}$.

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\tau|\pi_{\boldsymbol{\theta}})}[r(\tau)] \\ &= \nabla_{\boldsymbol{\theta}} \int r(\tau) p(\tau | \pi_{\boldsymbol{\theta}}) d\tau \\ &= \int r(\tau) \nabla_{\boldsymbol{\theta}} p(\tau | \pi_{\boldsymbol{\theta}}) d\tau \\ &= \int r(\tau) p(\tau | \pi_{\boldsymbol{\theta}}) \underbrace{\frac{\nabla_{\boldsymbol{\theta}} p(\tau | \pi_{\boldsymbol{\theta}})}{p(\tau | \pi_{\boldsymbol{\theta}})}}_{\nabla_{\boldsymbol{\theta}} \log p(\tau|\pi_{\boldsymbol{\theta}})} d\tau \\ &= \mathbb{E}_{p(\tau|\pi_{\boldsymbol{\theta}})}[r(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau | \pi_{\boldsymbol{\theta}})] \quad \square \end{aligned}$$

To arrive at a practicable algorithm, we can simplify the gradient inside the expectation on the right side of (3.4) even further by plugging in (3.1):

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log p(\tau | \pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} \log \left(\prod_{t=0}^{\infty} \pi_{\boldsymbol{\theta}}(a^{(t)} | s^{(t)}) \mathbb{1}_{\{r(s^{(t)}, a^{(t)})\}}(r^{(t)}) p(s^{(t+1)} | s^{(t)}, a^{(t)}) \right) \\ &= \sum_{t=0}^{\infty} \left(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a^{(t)} | s^{(t)}) + \underbrace{\nabla_{\boldsymbol{\theta}} \log \mathbb{1}_{\{r(s^{(t)}, a^{(t)})\}}(r^{(t)})}_{\text{constant w.r.t. } \boldsymbol{\theta}} + \underbrace{\nabla_{\boldsymbol{\theta}} \log p(s^{(t+1)} | s^{(t)}, a^{(t)})}_{\text{constant w.r.t. } \boldsymbol{\theta}} \right) \\ &= \sum_{t=0}^{\infty} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a^{(t)} | s^{(t)}). \quad (3.5) \end{aligned}$$

The expression now only requires differentiating $\pi_{\boldsymbol{\theta}}$, which is possible if a suitable parametric family with tractable derivatives has been chosen. An N -sample Monte Carlo estimator of (3.4) would now draw N trajectories $\tau_k \sim p(\tau | \pi_{\boldsymbol{\theta}})$ up to a large but finite time horizon T and obtain the gradient estimate

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{k=1}^N \sum_{t=0}^T r(\tau_k) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_k^{(t)} | s_k^{(t)}).$$

Putting everything together results in the gradient-based Algorithm 1 known as Monte Carlo Policy Gradient or REINFORCE (Williams, 1992). Note that the normalisation factor is subsumed by the learning rate α . To explain the name, assume there is only one trajectory and only two actions, $A = \{a, b\}$. If both appear with equal frequency in the sampled trajectory, no adaption is made. Now w.l.o.g. let a appear more frequently than b . If $r(\tau_1) > 0$, probability is shifted towards a , and if $r(\tau_1) < 0$, probability is shifted towards b . In this way, good actions become reinforced.

Algorithm 1: REINFORCE

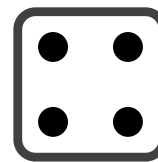
```

1 randomly initialise the policy parameters  $\theta$ 
2 for epoch = 1, 2 ... do
3   sample  $N$  trajectories  $\tau_1, \dots, \tau_N \sim p(\cdot \mid \pi_\theta)$ 
4    $\Delta\theta \leftarrow \mathbf{0}$ 
5   for  $k = 1, \dots, N$  do
6      $r(\tau_k) \leftarrow \sum_{t=0}^T \gamma^t r_k^{(t)}$ 
7     for  $t = 0, 1, \dots, T$  do
8        $\Delta\theta \leftarrow \Delta\theta + r(\tau_k) \nabla_\theta \log \pi_\theta(a_k^{(t)} \mid s_k^{(t)})$ 
9    $\theta \leftarrow \theta + \alpha \Delta\theta$ 

```

Chapter 4

Multi-agent Reinforcement Learning



In this chapter, we fuse the theories from Chapters 2 and 3 to model sequential decision-making processes with multiple agents. Let us take a step back and recall what each of the theories brings to the table.

- Game theory brings an interaction component by considering *multiple entities* (called players) who each have their own *decision options* (called actions) and a notion of *feedback signal* (called utility) and seek to find an optimal behavioural rule (called strategy).
- Reinforcement learning, on the other hand, brings a temporal component by considering different *environmental conditions* (called states) and a *progression mechanism* (called transitions) that a *single entity* (called agent) finds itself in, who has *decision options* (called actions) and a notion of *feedback signal* (called reward) and seeks to find an optimal *behavioural rule* (called policy).

It is clear that both theories share several components that are just called differently and which we summarise in Table 3. The most salient shared feature is the decision options, which are even called the same in both theories. These decisions lead to feedback, with the only difference between utility and reward being that the latter additionally depends on the state. In both theories, the goal consists in determining a suitable behaviour, where again the only difference between strategy and policy is that the latter is conditioned on a state. However, the unique selling points are that game theory considers *multiple* entities which introduces interaction, whereas reinforcement learning considers *state transitions* which introduce sequentiality.

game theory	$\longleftrightarrow$	RL
player	$\longleftrightarrow$	agent
action	$\longleftrightarrow$	action
utility	$\longleftrightarrow$	reward
strategy	$\longleftrightarrow$	policy

Table 3: Terminology translation between game theory and RL.

Combining the two theories is straightforwardly done by merging all the components while giving precedence to the more general version in case of conflicts (e.g., single-agent reward vs. multi-player utility). The result, quite intuitively, is a generalisation of MDPs (Definition 10) where at every state a normal-form game is played.

Definition 12 (Stochastic Game / Multi-Agent MDP). A *Stochastic Game* or *Multi-Agent MDP* is a 6-tuple $(N, S, (A_i)_{i \in N}, p, (r_i)_{i \in N}, \gamma)$, consisting of

- a finite set of *agents* $N = \{1, \dots, n\}$
- a finite set of *states* S ,
- an *action* set A_i per agent $i \in N$, collected in $A := A_1 \times \dots \times A_n$,
- a conditional *transition probability* distribution $p : S \times A \rightarrow \Delta(S)$,
- a *reward* function $r_i : S \times A \rightarrow \mathbb{R}$ per agent $i \in N$ which is bounded, and
- a *discount factor* $\gamma \in [0, 1)$.

The differences to the MDP Definition 10 are highlighted in red: indices for the agents are added, there are multiple action sets, multiple reward functions, and the transition probabilities and reward functions now depend on the actions of *all* agents. The latter is a rather strong assumption which can be weakened by the introduction of partial observability, but this is out of the scope of this work.

A stochastic game progresses analogously to an MDP: given state $s^{(t)} \in S$ at time t , each agent simultaneously chooses an action $a_i^{(t)} \in A_i$, observes the actions $\mathbf{a}^{(t)}$ of all agents, receives an immediate personal reward $r_i(s^{(t)}, \mathbf{a}^{(t)})$, and then the game transitions to a new state $s^{(t+1)}$ with probability $p(s^{(t+1)} \mid s^{(t)}, \mathbf{a}^{(t)})$. It is crucial to emphasise once again that the agents select their actions *simultaneously*, just like in normal-form games. This selection is governed by their individual policy.

Definition 13 (Multi-agent Policy). A *multi-agent policy* of an agent $i \in N$ is any function $\pi_i : S \rightarrow \Delta(A_i)$ that maps a state to a Multinoulli distribution over actions of agent i .

Again, like in the single-agent setting, we seek an optimal policy for each agent that maximises its expected discounted sum of rewards. A straightforward and standard approach is *independent RL*: simply apply any single-agent RL algorithm for each agent independently, thus treating the other agents as part of the environment. Algorithm 2 shows this for the REINFORCE algorithm, with the differences to Algorithm 1 highlighted in red.

Algorithm 2: independent-REINFORCE

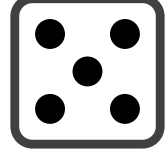
```

1 randomly initialise the policy parameters  $\theta_1, \dots, \theta_n$ 
2 for epoch = 1, 2 ... do
3   sample  $N$  trajectories  $\tau_1, \dots, \tau_N \sim p(\cdot \mid \pi_{\theta_1}, \dots, \pi_{\theta_n})$ 
4    $\Delta\theta_i \leftarrow \mathbf{0}$  for each agent  $i \in N$ 
5   for  $k = 1, \dots, N$  do
6      $r_i(\tau_k) \leftarrow \sum_{t=0}^T \gamma^t r_{i,k}^{(t)}$  for each agent  $i \in N$ 
7     for  $t = 0, 1, \dots, T$  do
8       for  $i \in N$  do
9          $\Delta\theta_i \leftarrow \Delta\theta_i + r_i(\tau_k) \nabla_{\theta_i} \log \pi_{\theta_i}(a_{i,k}^{(t)} \mid s_{i,k}^{(t)})$ 
10   $\theta_i \leftarrow \theta_i + \alpha_i \Delta\theta_i$  for each agent  $i \in N$ 

```

However, in the multi-agent setting, applying single-agent RL methods independently no longer directly corresponds to maximising the expected discounted reward since the optimal multi-agent policy additionally depends on the opponents' policies. This crucial issue is commonly referred to as the *non-stationarity problem* of MARL. Leaving the opponents unaddressed can lead to undesired final results or even algorithm divergence, as we will see in the next chapter.

Learning with Opponent-Learning Awareness



In this chapter, I put all concepts and insights from Chapters 2 to 4 together to describe a state-of-the-art multi-agent RL algorithm called Learning with Opponent-Learning Awareness (LOLA; Foerster et al., 2018), which does not simply treat the other agents as a static part of the environment, in Section 5.1, and discuss empirical evaluations on simple partially competitive settings in Section 5.2.

5.1 The Algorithm

As discussed in Chapter 4, independent RL methods can have significant drawbacks in non-cooperative settings. We will see this quite drastically in the experiments in the next section. First, I reformulate the independent policy gradient approach from Section 3.2 to then introduce LOLA. For simplicity, I do so for the case of $n = 2$ agents and give all equations from the perspective of agent 1. This is w.l.o.g., due to the symmetry of the derivation in an arbitrary number of agents.

Denote by θ_1, θ_2 the agents' policy parameters and by $J_i(\theta_1, \theta_2)$ again the expected discounted sum of rewards for agent i . Naïve independent policy gradient ascent (IPGA) for agent i with step size (or “learning rate”) $\alpha_i > 0$ would proceed as

$$\theta_i \leftarrow \theta_i + \alpha_i \mathbf{f}_i^{\text{IPGA}}(\theta_1, \theta_2), \quad \text{where } \mathbf{f}_i^{\text{IPGA}}(\theta_1, \theta_2) := \nabla_{\theta_i} J_i(\theta_1, \theta_2). \quad (5.1)$$

Taking the perspective of agent 1 from now on, the update rule (5.1) completely ignores the fact that agent 2 is learning, i.e., updating its own policy just as well! This is where LOLA steps in: under the assumption that agent 2 too performs an independent policy gradient ascent update $\Delta\theta_2 = \alpha_2 \mathbf{f}_2^{\text{IPGA}}(\theta_1, \theta_2)$, agent 1 can account for this learning process by “looking ahead”.

Insight. *Instead of maximising $J_1(\theta_1, \theta_2)$, agent 1 should maximise $J_1(\theta_1, \theta_2 + \Delta\theta_2)$.*

Observe that $\Delta\theta_2$ depends on θ_1 as well, and therefore agent 1 now needs to differentiate through the opponents' policy update. Before we start the derivation, note that *I use the numerator layout** for derivatives, i.e., the gradient is a row vector and the Jacobian

*see https://en.wikipedia.org/wiki/Matrix_calculus#Layout_conventions

a matrix of gradient rows. The multivariate chain rule then gives us

$$\begin{aligned}
\nabla_{\theta_1} J_1(\theta_1, \theta_2 + \Delta\theta_2) &= \underbrace{\nabla_{(\omega_1, \omega_2)} J_1(\omega_1, \omega_2)}_{\text{gradient vector}} \bigg|_{\substack{\omega_1 = \theta_1 \\ \omega_2 = \theta_2 + \Delta\theta_2}} \underbrace{\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ & & & \alpha_2 \nabla_{\theta_1} f_2^{\text{IPGA}}(\theta_1, \theta_2) \end{bmatrix}}_{\text{Jacobi matrix}} \\
&= \nabla_{\omega_1} J_1(\omega_1, \omega_2) \bigg|_{\substack{\omega_1 = \theta_1 \\ \omega_2 = \theta_2 + \Delta\theta_2}} + \alpha_2 \nabla_{\omega_2} J_1(\omega_1, \omega_2) \bigg|_{\substack{\omega_1 = \theta_1 \\ \omega_2 = \theta_2 + \Delta\theta_2}} \nabla_{\theta_1} f_2^{\text{IPGA}}(\theta_1, \theta_2)
\end{aligned} \tag{5.2}$$

To simplify the gradient evaluations, we assume that $\Delta\theta_2$ is sufficiently small, which can be achieved by choosing a small learning rate α_2 , and set $\omega_2 = \theta_2$. The modified gradient update function for agent 1 to be plugged into (5.1) is then

$$f_1^{\text{LOLA}}(\theta_1, \theta_2) := \underbrace{\nabla_{\theta_1} J_1(\theta_1, \theta_2)}_{\text{vanilla gradient}} + \alpha_2 \nabla_{\theta_2} J_1(\theta_1, \theta_2) \underbrace{\nabla_{\theta_1} \nabla_{\theta_2} J_2(\theta_1, \theta_2)}_{\text{LOLA Hessian}}. \tag{5.3}$$

This derivation is my own contribution as Foerster et al. do it via first-order Taylor approximation, which is based on the same small-update assumption after all. Comparing (5.3) with (5.1), we see there is an additional term that accounts for the impact of agent 1's policy on the anticipated parameter update of agent 2.

Since obtaining the exact gradient and Hessian of the objective function is intractable for most practical purposes, we again seek to replace them with Monte Carlo estimates. The estimator for the gradients $\nabla_{\theta_1} J_1(\theta_1, \theta_2)$ and $\nabla_{\theta_2} J_1(\theta_1, \theta_2)$ is the exact same as in Section 3.2 based on Theorem 1. The estimator for the Hessian $\nabla_{\theta_1} \nabla_{\theta_2} J_2(\theta_1, \theta_2)$ requires a rather tedious derivation that, however, closely resembles the proof of the policy gradient theorem.

Theorem 2 (LOLA Hessian, Foerster et al. (2018)). *The Hessian in (5.3) is Monte Carlo-estimable as the following equality holds:*

$$\nabla_{\theta_1} \nabla_{\theta_2} J_2(\theta_1, \theta_2) = \mathbb{E}_{p(\tau | \pi_{\theta_1}, \pi_{\theta_2})} \left[r_2(\tau) \left(\nabla_{\theta_1} \log p(\tau | \pi_{\theta_1}) \right)^\top \left(\nabla_{\theta_2} p(\tau | \pi_{\theta_2}) \right) \right].$$

Proof. Omitted. See Foerster et al., Appendix A.1. □

Following the same simplification steps as in (3.5), we arrive at the following N -sample Monte Carlo estimator with subscript k indexing the sample trajectory:

$$\nabla_{\theta_1} \nabla_{\theta_2} J_2(\theta_1, \theta_2) \approx \frac{1}{N} \sum_{k=1}^N \sum_{t=0}^T r_2(\tau_k) \left(\sum_{t'=0}^t \nabla_{\theta_1} \log \pi_{\theta_1}(a_{1,k}^{(t')} | s_k^{(t')}) \right)^\top \left(\sum_{t'=0}^t \nabla_{\theta_2} \log \pi_{\theta_2}(a_{2,k}^{(t')} | s_k^{(t')}) \right).$$

Note that the original LOLA estimator additionally applies advanced techniques for variance reduction, such as causality and baseline subtraction. For simplicity of presentation, I do not consider these in this essay and stick with the vanilla REINFORCE-style version of the LOLA algorithm, which is given in Algorithm 3. The difference to the independent REINFORCE Algorithm 2 consists in the gradient update and is highlighted in red.

Algorithm 3: LOLA-REINFORCE

```

1 randomly initialise the policy parameters  $\theta_1, \dots, \theta_n$ 
2 for epoch = 1, 2 ... do
3   sample  $N$  trajectories  $\tau_1, \dots, \tau_N \sim p(\cdot \mid \pi_{\theta_1}, \dots, \pi_{\theta_n})$ 
4    $\Delta\theta_i \leftarrow \mathbf{0}$  for each agent  $i \in N$ 
5   for  $k = 1, \dots, N$  do
6      $r_i(\tau_k) \leftarrow \sum_{t=0}^T \gamma^t r_{i,k}^{(t)}$  for each agent  $i \in N$ 
7     for  $t = 0, 1, \dots, T$  do
8       for  $i \in N$  do
9          $\Delta\theta_i \leftarrow \Delta\theta_i + r_i(\tau_k) \nabla_{\theta_i} \log \pi_{\theta_i}(a_{i,k}^{(t)} \mid s_{i,k}^{(t)})$ 
10        for  $j \in N \setminus \{i\}$  do
11           $\Delta\theta_i \leftarrow \Delta\theta_i + r_j(\tau_k) \left( \sum_{t'=0}^t \nabla_{\theta_i} \log \pi_{\theta_i}(a_{i,k}^{(t')} \mid s_k^{(t')}) \right)^\top$ 
12           $\cdot \left( \sum_{t'=0}^t \nabla_{\theta_j} \log \pi_{\theta_j}(a_{j,k}^{(t')} \mid s_k^{(t')}) \right)$ 
13    $\theta_i \leftarrow \theta_i + \alpha_i \Delta\theta_i$  for each agent  $i \in N$ 

```

In settings where the assumption of having access to the exact parameters of the opponents no longer holds, LOLA can be extended by *opponent modelling*. The opponent parameters are estimated from the interaction via maximum likelihood and replace the exact parameters in the update rule, with the rest of the algorithm remaining the same. I refer to Foerster et al. for details.

5.2 Emergent Behaviour

To investigate how opponent-learning awareness affects the emergent behaviour of the agents, Foerster et al. conducted empirical experiments on different environments. To this end, a suitable testbed is provided by repeated games (cp. Section 2.2) based on the two-player normal-form games from Examples 1 and 2 in Section 2.1.

Recall that, in a repeated game, agents condition their actions on past history with finite memory size, which violates the Markov assumption of only depending on the current state in stochastic games. However, Press & Dyson (2012) prove that agents with a good Markovian strategy, i.e., one that only considers actions of the last step, can effectively force the game to be played as if it was Markovian in the first place. Hence, the construction of a stochastic game for a repeated game is as follows:

Definition 14 (Repeated-Game Multi-Agent MDP). A two-player normal-form game $(N, (A_1, A_2), (u_1, u_2))$ can be extended to a repeated-game multi-agent MDP via

- $S := (A_1 \times A_2) \cup \{\emptyset\}$,
- $p(s' \mid s, (a_1, a_2)) := \mathbb{1}\{s' = (a_1, a_2)\}$ for all $s \in S$, and
- $r_i(s, (a_1, a_2)) := u_i(a_1, a_2)$, the (a_1, a_2) -entry in the payoff matrix.

The game then evolves by starting in the empty state $s^{(0)} \leftarrow \emptyset$ and deterministically progressing as $s^{(t+1)} \leftarrow (a_1^{(t)}, a_2^{(t)})$ for $t \in \mathbb{N}_0$, i.e., the state tracks each agent’s last action. Since all the games we consider have two agents with two actions each, their associated stochastic game has $1 + 2 \cdot 2 = 5$ states.

Foerster et al. train agents on Repeated Matching Pennies (RMP) and the Repeated Prisoner’s Dilemma (RPD) using IPGA and LOLA with both exact gradient ascent and Monte-carlo gradient ascent using a step size of $\alpha_i = 0.005$, and discount factors $\gamma = 0.9$ for RMP and $\gamma = 0.96$ for RPD, respectively. They empirically choose γ as the largest possible value that still results in stable training. In the following, I discuss the results they obtained after 50 independent training runs with different seeds.

RMP. Figure 1 shows that IPGA fails to converge as the learnt heads probability varies drastically across training runs (cp. Figure 1a) and the average per-step reward during the repeated game exhibits a high variance (cp. Figure 1c). In contrast, LOLA quite clearly learns a 50/50 strategy for both agents (cp. Figure 1b), which corresponds to the unique Nash equilibrium as per Lemma 1 and is the optimal strategy in this zero-sum game. Using Monte Carlo-estimated gradients (bottom row) instead of exact ones (top row) only introduces noise, but the overall observations are the same.

RPD. Figure 2 shows that IPGA at least does not diverge but converges to a sub-optimal always-defect (DD, Definition 8) strategy (cp. Figure 2a) with long-term payoff -2 for both agents (cp. Figure 2c), while LOLA converges to a strategy reminiscent of the tit-for-tat (TFT, Definition 9) strategy (cp. Figure 2b) with long-term payoff around -1.2 for both agents (cp. Figure 2c), which is close to the social welfare optimum of -1 . Again, Monte Carlo gradients (bottom row) lead to noisier results than exact gradients (top row) but still afford the same conclusions.

In summary, we can make the following statements:

1. In RMP, LOLA enables convergence of policy gradient ascent in the first place, which does not happen with IPGA.
2. In RPD, LOLA leads to emergent reciprocity and cooperation as it improves the learnt strategy from DD to something close to TFT. The remarkable fact is that, nowhere in the LOLA algorithm, social welfare is actively incorporated but instead *“LOLA agents learn to cooperate out of self-interest”* (Foerster et al.).

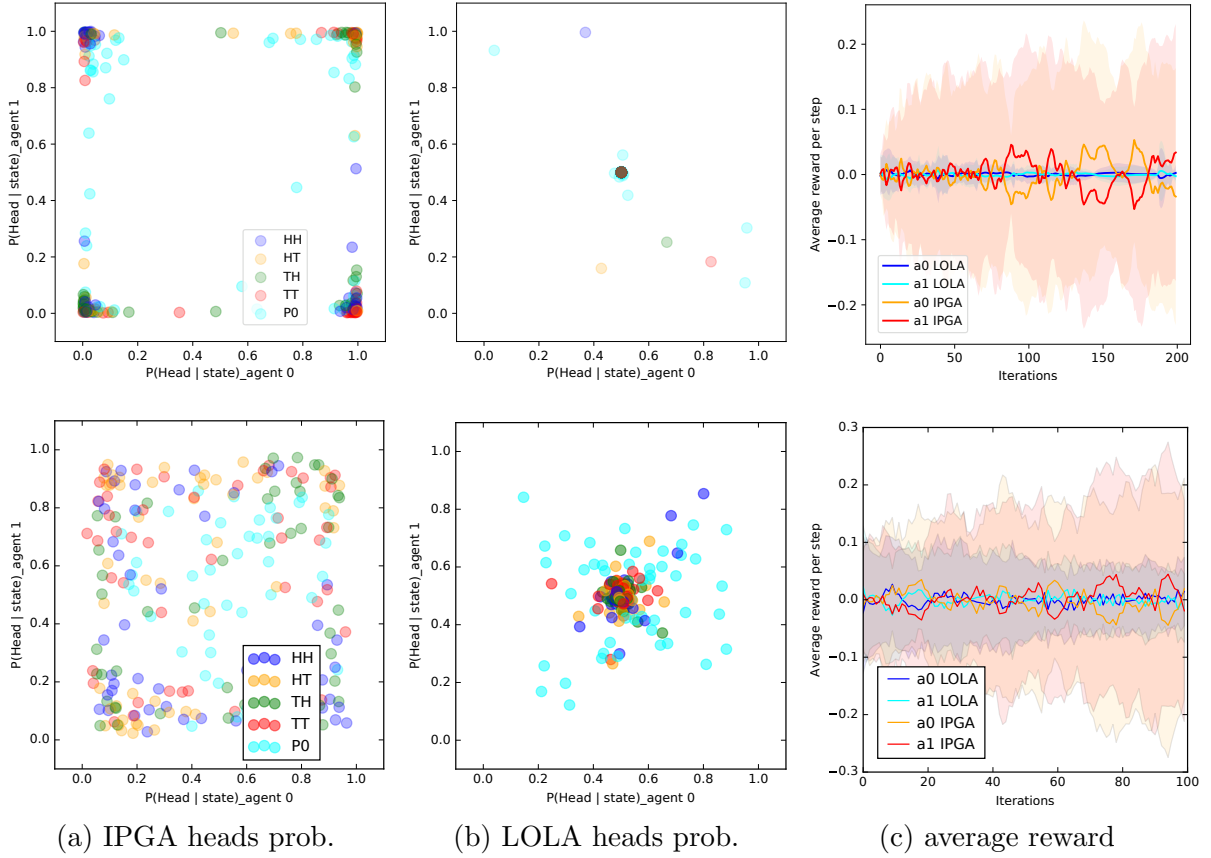


Figure 1: Comparison of IPGA and LOLA on RMP with exact gradients (top) and Monte Carlo-estimated gradients (bottom). In the left column, **state** = P0 refers to the initial state and **state** $\in \{\text{HH}, \text{HT}, \text{TH}, \text{TT}\}$ to the previous state where the respective action combination was played (H = heads, T = tails). Adapted from Foerster et al.

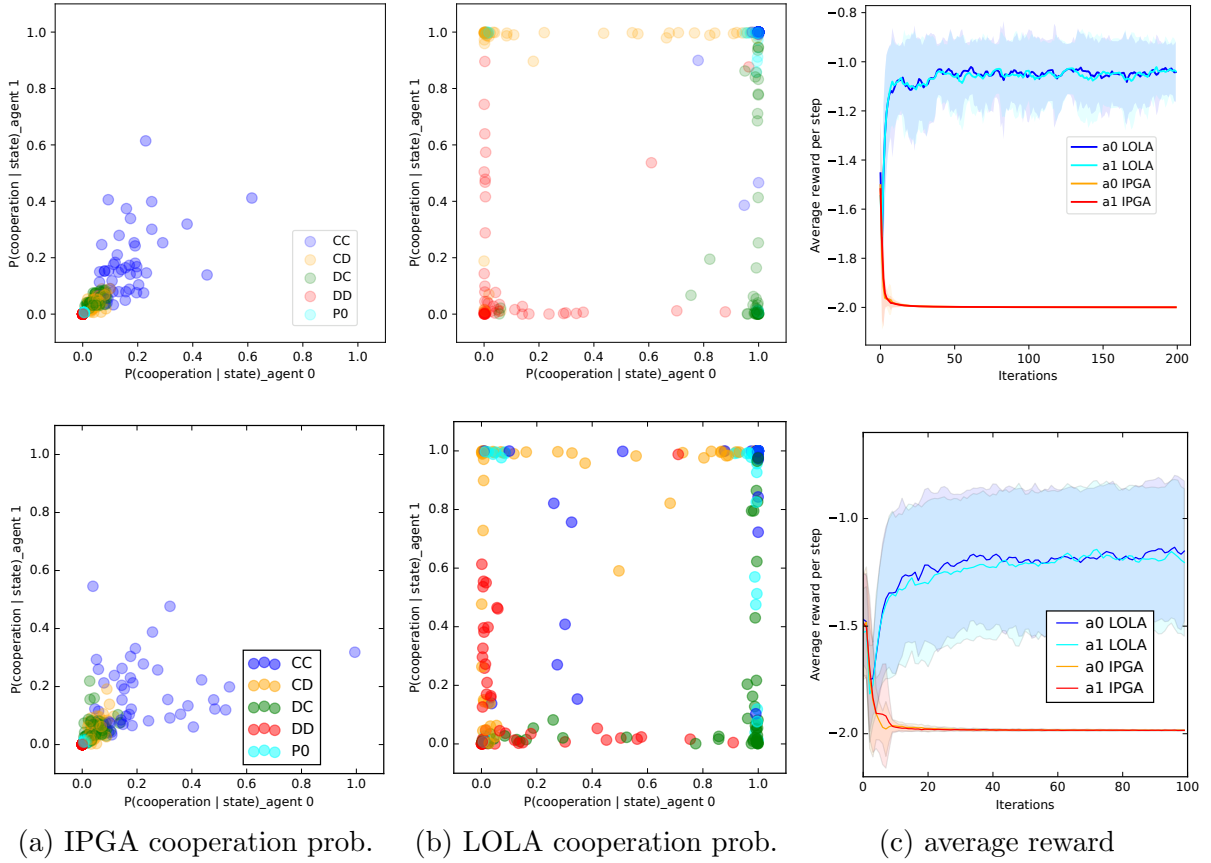
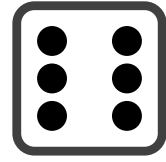


Figure 2: Comparison of IPGA and LOLA on RPD with exact gradients (top) and Monte Carlo-estimated gradients (bottom). In the left column, **state** = P0 refers to the initial state and **state** $\in \{\text{CC}, \text{CD}, \text{DC}, \text{DD}\}$ to the previous state where the respective action combination was played (C = cooperate, D = defect). Adapted from Foerster et al.

Chapter 6

Conclusion

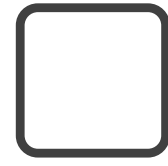


In this essay, I demonstrated, with carefully chosen notation and structure, how the distinct fields of game theory and reinforcement learning unite to become multi-agent reinforcement learning, a crucial framework moving forward with regard to future AI systems. Game theory brings the strategic interaction component, which is vital for agents to take into account other intelligent actors. In contrast, vanilla reinforcement learning brings the sequential state transition component, which allows dynamic interactions with a complex environment.

After explaining the theory in detail, I discussed multi-agent RL algorithms with a particular focus on convergence and social welfare. While independent policy gradient ascent, a standard approach, is straightforward to implement, it ignores the learning capabilities of other agents. As a result, it leads to divergence in repeated Matching Pennies games and a suboptimal always-defect strategy in the repeated Prisoner's Dilemma. These challenges can be overcome by LOLA, which explicitly differentiates through the opponents' policy updates to account for their learning process. This simple but clever correction leads to convergence in RMP and a socially optimal tit-for-tat strategy in RPD. Remarkably, this reciprocity and cooperation emerges purely out of the agents' own self-interest, without any enforcement in the algorithm design. However, it comes at the cost of a more expensive gradient computation.

On a meta-level, my essay showcases how the harmonisation of two rather distinct theories can improve the understanding of a complex framework like multi-agent RL, and how a seemingly simple but clever algorithmic tweak can drastically affect the emergent behaviour of an AI system. With AI having become a highly specialised research area in recent years, it is my firm belief that the community would benefit from more cross-cutting disquisitions like this essay.

Bibliography



-
-
- Axelrod, R. and Hamilton, W. D. The Evolution of Cooperation. *Science*, 211(4489): 1390–1396, March 1981.
- Bellman, R. The Theory of Dynamic Programming. *Bulletin of the American Mathematical Society*, 60(6):503 – 515, 1954. Publisher: American Mathematical Society.
- Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of Computing*, pp. 71–78, Seattle WA USA, May 2006. ACM.
- Flood, M. M. *Some Experimental Games*. RAND Corporation, Santa Monica, CA, 1952.
- Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with Opponent-Learning Awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18*, pp. 122–130, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems. event-place: Stockholm, Sweden.
- Matignon, L., Laurent, G. J., and Le Fort-Piat, N. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, February 2012.
- Minsky, M. L. *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain Model Problem*. PhD Thesis, 1954.
- Myerson, R. B. *Game Theory: Analysis of Conflict*. Harvard University Press, March 2013.
- Nash, J. Non-Cooperative Games. *The Annals of Mathematics*, 54(2):286, September 1951.
- Neumann, J. V. and Morgenstern, O. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, USA, 1944.
- Press, W. H. and Dyson, F. J. Iterated Prisoner’s Dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413, June 2012.

- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 1998.
- Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S., Leen, T., and Müller, K. (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.
- Thorndike, E. L. Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i–109, 1898.
- Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019.
- Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.
- Zermelo, E. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the fifth international congress of mathematicians*, volume 2, pp. 501–504. Cambridge University Press Cambridge, 1913.