Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning in Games: First Steps and Challenges

Stefano V. Albrecht, Filippos Christianos, Lukas Schäfer
Slides by: Leonard Hinckeldey

Multi-Agent Reinforcement Learning: Foundations and Modern Approaches

Stefano V. Albrecht, Filippos Christianos, Lukas Schäfer

Published by MIT Press
(print version will be released on 17 December 2024)

This lecture is based on *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches* by Stefano V. Albrecht, Filippos Christianos and Lukas Schäfer

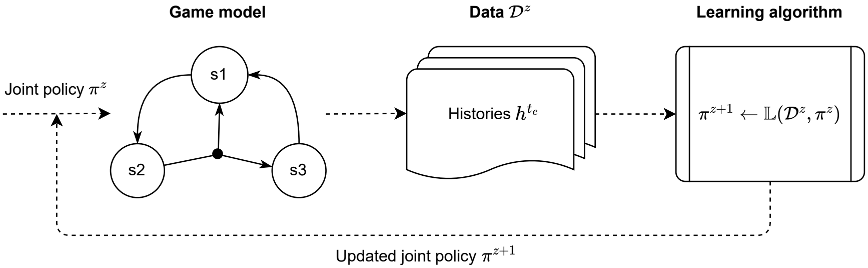
The book can be downloaded for free at www.marl-book.com.

Lecture Outline

- Learning framework for MARL
- Independent learning
- Central learning
- Modes of learning
- Challenges of MARL

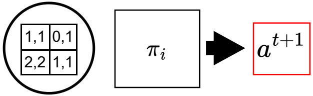
The MARL learning framework

MARL Learning Process



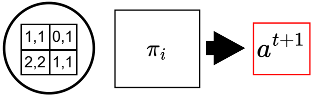
- The game model defines agent interaction
- Data of these interactions are collected as $\mathcal{D}^z = \{h^{t_e} \mid e = 1, \ldots, z\}, z \geq 0$
- A learning algorithm updates each agent's policy $\pi^{z+1} = \mathbb{L}(\mathcal{D}^z, \pi^z)$
- The learning goal is a chosen solution concept, e.g. Nash equilibrium

Game Model's and Joint Policy

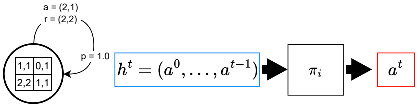


Normal-form games

Game Model's and Joint Policy

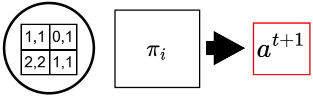


Normal-form games

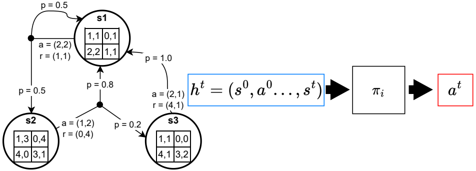


Repeated normal-form games

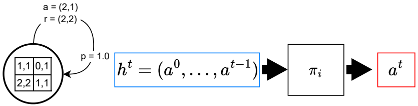
Game Model's and Joint Policy



Normal-form games

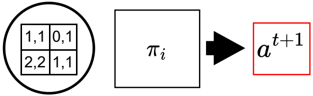


Stochastic games

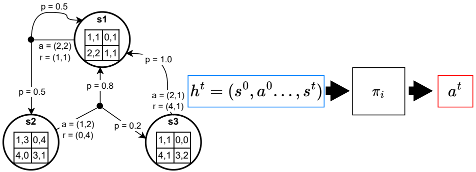


Repeated normal-form games

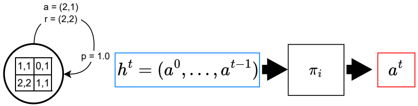
Game Model's and Joint Policy



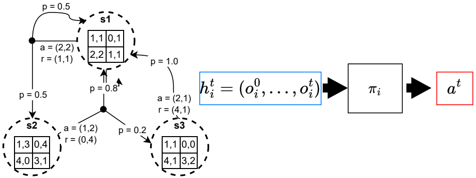
Normal-form games



Stochastic games



Repeated normal-form games



Partially observable stochastic games

Convergence

To evaluate the learning algorithm, we typically assess whether the learnt joint policy has **converged** to the optimal joint policy.

$$\lim_{z \to \infty} \pi^z = \pi^*$$

- The optimal joint policy may differ depending on the chosen game solution
- There may be many valid solutions, e.g., multiple Nash equilibria
- In practice, we cannot collect infinite data
- Learning typically stops after a predefined 'budget' (e.g. training steps) is reached or changes in the policy are below a predefined threshold

Single-Agent RL Reductions

The simplest way to apply reinforcement learning algorithms in multi-agent settings is to reduce them to single-agent problems.

- We can apply single-agent RL algorithms to each agent independently
  - This is known as **independent** learning; agents do not explicitly consider each other instead they treat other agents as part of the environment
- Alternatively, we can apply one single-agent RL to all agents centrally
  - This is referred to as **central** learning, where a central policy is learned, providing action probability across all agents' actions

Central Learning

Central Learning

In the central learning framework, we learn a single central policy π_c, which receives observations of all agents and selects an action for each agent.

- Requires transforming the joint reward $(r_1, \ldots, r_n)$, into a single scalar reward r
- This can be easy in some settings, e.g. in games with common rewards $r = r_i$, but difficult in zero-sum or general-sum games
- Central learning does not scale well with the number of agents as the joint action space grows exponentially with the number of agents
- This framework may also not be suitable in environments that require agents to act independently based on localized observations

Central Q-Learning

Algorithm Central Q-learning

Initialize: $Q(s, a) = 0$ for all $s \in S$ and $a \in A = A_1 \times \ldots \times A_n$
Repeat for every episode:
for $t = 0, 1, 2, \ldots$ **do**
    Observe current state s^t
    With probability ϵ: choose random joint action $a^t \in A$
    Otherwise: choose joint action $a^t \in \arg\max_a Q(s^t, a)$
    Apply joint action a^t, observe rewards $r_1^t, \ldots, r_n^t$ and next state s^{t+1}
    Transform $r_1^t, \ldots, r_n^t$ into scalar reward r^t
    $Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[r^t + \gamma \max_{a'} Q(s^{t+1}, a') - Q(s^t, a^t)]$

Independent Learning

Independent Learning

In the independent learning framework, each agent i learns its policy π_i using only its local history of observations, treating the effects of other agents' actions as part of the environment.

- From the perspective of the individual agent, the environment transition function looks like this:

$$\mathcal{T}_i(s^{t+1}|s^t, a_i) \propto \sum_{a_{-i} \in A_{-i}} \mathcal{T}(s^{t+1}|s^t, \langle a_i, a_{-i} \rangle) \prod_{j \neq i} \pi_j(a_j|s^t)$$

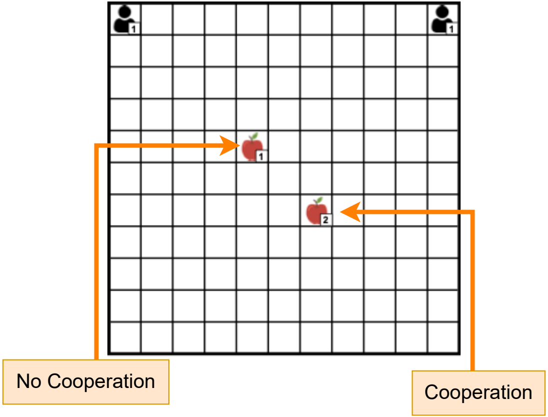
- As each agent j's policies are updated, the action probabilities π_j change
- From agent i's perspective its transition function $\mathcal{T}_i$ thus appears to be non-stationary

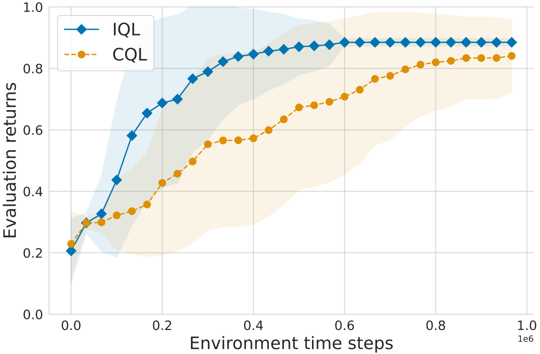
Independent Q-learning

Algorithm Independent Q-learning (IQL) for stochastic games

// Algorithm controls agent i
Initialize: $Q_i(s, a_i) = 0$ for all $s \in S$, $a_i \in A_i$
Repeat for every episode:
for $t = 0, 1, 2, \ldots$ **do**
    Observe current state s^t
    With probability ϵ: choose random action $a_i^t \in A_i$
    Otherwise: choose action $a_i^t \in \arg\max_{a_i} Q_i(s^t, a_i)$
    (meanwhile, other agents $j \neq i$ choose their actions a_j^t)
    Observe own reward r_i^t and next state s^{t+1}
    $Q_i(s^t, a_i^t) \leftarrow Q_i(s^t, a_i^t) + \alpha[r_i^t + \gamma \max_{a_i'} Q_i(s^{t+1}, a_i') - Q_i(s^t, a_i^t)]$

IQL and CQL in Level-based Foraging





- IQL can learn more quickly as CQL needs to explore $6^2 = 36$ actions in each state

Modes of Operation in MARL

In addition to independent and central learning, there are different modes of learning algorithms. Algorithms can use **self-play** or **mixed-play**.

Self-play:

- Refers to two related but distinct modes of operation in MARL, **algorithm self-play** and **policy self-play**
- **Algorithm self-play** assumes that all agents use the same learning algorithm
- **Policy self-play** is more literal and means that an agent's policy is trained directly against itself

Mixed-play

- Refers to instances where agents use different learning algorithms

MARL Challenges

MARL Challenges

MARL algorithms inherit issues from single-agent RL, such as:

- Unknown environment dynamics
- Exploration-exploitation dilemma
- Non-stationarity from bootstrapping
- Temporal credit assignment

In addition to this, MARL algorithms face challenges that arise from learning in a dynamic multi-agent system.

Non-Stationarity

A stochastic process ${X^t}_{t \in \mathbb{N}^0}$ is stationary if:

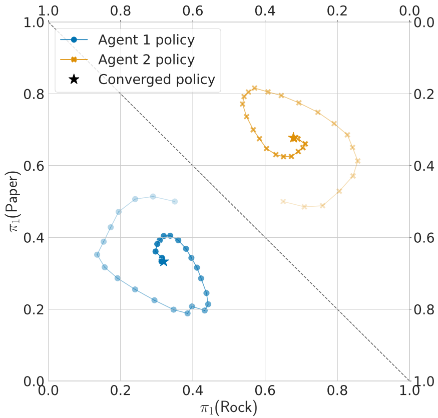
- The probability distribution of $X^{t+\tau}$ does not depend on $\tau \in \mathbb{N}^0$, where t and $t + \tau$ are time indices
- This means that the dynamics of the process do not change over time

Now X^t samples the state s^t at each time step t:

- In a MDP, X^t is completely defined by the state transition function $\mathcal{T}(s^t|s^{t-1}, a^{t-1})$ and the agents policy π which selects an action $a \sim \pi(.|s)$
- The **Markov property** means that this property is stationary in that s^t depends only on s^{t-1}, a^{t-1} and a^{t-1} depends only on s^{t-1} via $\pi(.|s^{t-1})$
- In RL, the policy does, however, change over time through the learning process $\pi^{z+1} = \mathbb{L}(\mathcal{D}^z, \pi^z)$, which leads to **non-stationarity** in X^t

Non-Stationarity in Multi-Agent Settings

In MARL, non-stationarity is exacerbated by multiple agents changing their policies over time.



- $\pi^{z+1} = \mathbb{L}(\mathcal{D}^z, \pi^z)$ updates an entire joint policy $\pi^z = (\pi_1^z, \ldots, \pi_n^z)$
- Leads the entire environment to seem non-stationarity from each agent's perspective
- Can cause cyclic learning dynamics where agents co-adapt to each other's changing policies

Equilibrium Selection

A game might have multiple equilibrium solutions, each of which might yield different expected returns to the agents in the game. This leads to the problem of **equilibrium selection**

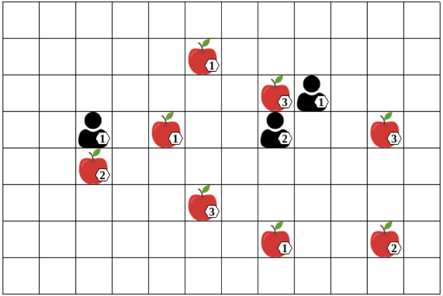
- Consider the stag hunt game
- Two hunters choose: cooperate to hunt stag (S) or go solo for hare (H)
- Nash equilibria: reward-dominant (S,S) maximizes reward, risk-dominant (H,H) minimizes risk
- (S,S) requires trust; (H,H) offers a safe, lower reward

	S	H
S	4,4	0,3
H	3,0	2,2

- Independent Q-learning may bias towards (H,H) due to initial action uncertainty
- Early random actions can reinforce (H,H) since deviating from H is penalized if the other agent chooses H

Multi-Agent Credit Assignment

Credit assignment in single-agent RL refers to determining which past action contributes to receiving rewards. In MARL, determining which agent has contributed to receiving rewards is an additional challenge.



- At time step t all agents simultaneously perform *collect*, each receiving reward $+1$
- Whose action led to the reward?
- The agent on the left did not contribute
- For a learning agent which only observes s^t, a^t, r^t, s^{t+1}, disentangling the action contributions is difficult

Joint Actions for Addressing Multi-Agent Credit Assignment

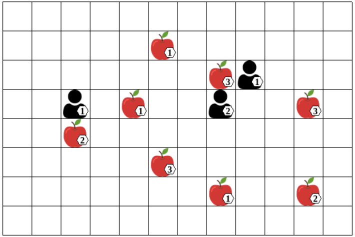
Joint actions can help disentangle agent contributions. Consider the RPS game:

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

1. Agents choose actions $(a_1, a_2) = (R, S)$ with agent 1 receiving reward $+1$
2. Agents choose $(a_1, a_2) = (R, P)$ with agent 1 receiving reward -1
3. Using $Q(s, a_1)$ does not model agent 2's action's effects and thus the value for action R appears to be 0
4. $Q_1(s, a_1, a_2)$ can represent the effect of agent 2 thus ascribing different values to joint action (R, S) and (R, P)

Scaling to Many Agents

The ability to scale to many agents is an important goal in MARL. Scaling agents is, however, a significant challenge for MARL as:



- # of **joint action** can grow **exponentially** with # agents since $|A| = |A_1| \cdot \ldots \cdot |A_n|$
- Changing # agents from 3 to 5 increases the number of joint actions from 216 to 7776
- If agents have associated features in s (e.g. agent position) then $|S|$ also increases exponentially
- In CL, this increases the decision problem, while in IL this increases issues of non-stationarity and multi-agent credit assignment

Summary

We covered:

- MARL Learning Process
- Independent and central learning
- Modes of learning in MARL
- Challenges of MARL

Next we'll cover:

- Foundational algorithms in MARL