Reinforcement Learning

Introduction to Multi-Agent Reinforcement Learning

Michael Herrmann, David Abel

Based on slides by Stefano V. Albrecht, Filippos Christianos, Lukas Schäfer, and Leonard Hinckeldey

THE UNIVERSITY *of* EDINBURGH informatics

Lecture Outline

- Multi-agent systems
- Game models
- Solution concepts
- Refinement concepts
- MARL learning framework
- Agent modeling
- Deep MARL
- Agent modeling with deep learning
- Outlook

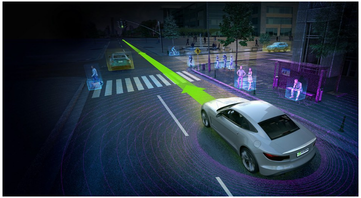
Introduction: Multi-agent systems

What is MARL?

Multi-agent reinforcement learning (MARL) is about finding optimal decision policies for two or more artificial agents interacting in a shared environment.

- Applying reinforcement learning (RL) algorithms to multi-agent systems
- Goal is to learn optimal policies for two or more agents

Computer games



Autonomous driving

Multi-robot warehouses

Automated trading

Mutli-Agent Systems

A multi-agent system consists of:

- **Environment:** The environment is a physical or virtual world whose state evolves and is influenced by the agents' actions within the environment.

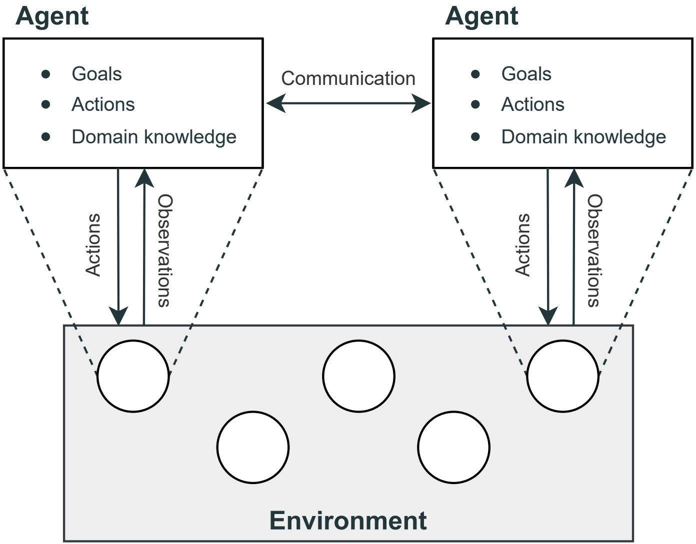
Mutli-Agent Systems

A multi-agent system consists of:

- **Environment:** The environment is a physical or virtual world whose state evolves and is influenced by the agents' actions within the environment.
- **Agents:** An agent is an entity which receives information about the state of the environment and can choose actions to influence the state.

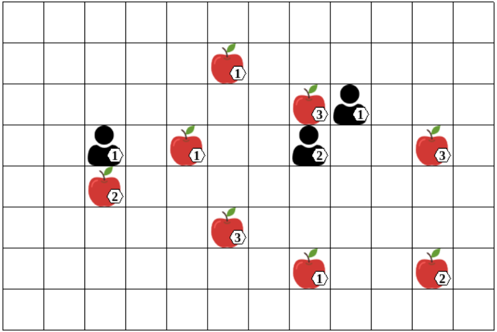
  $\Rightarrow$ Agents are goal-directed, e.g. maximizing returns

Multi-Agent Systems

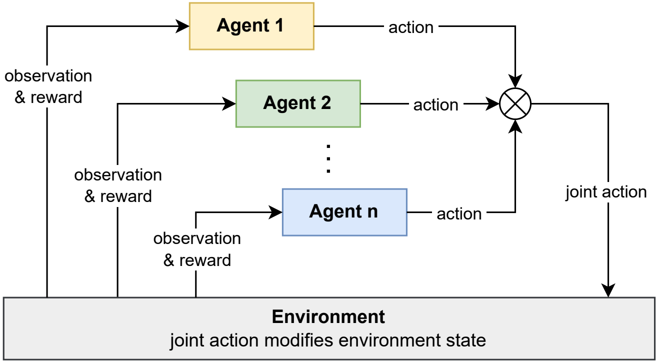


Example: Level-Based Foraging

- Three agents (robots) with varying skill levels
- Goal: to collect all items (apples)
- Items can be collected if a group of one or more agents are located next to the item and the sum of agents' levels is greater than or equal to the item level
- Action space $A = \{up, down, left, right, collect, noop\}$



MARL for Solving Multi-Agent Systems



- **Goal:** learn optimal policies for a set of agents in a multi-agent system
- Each agent chooses an action based on its policy $\Rightarrow$ joint action
- Joint action affects environment state; agents get rewards + new observations

Why MARL?

Why should we use MARL to find optimal solutions to multi-agent systems rather than controlling multiple 'agents' using a single-agent RL (SARL) algorithm?

Why MARL?

Why should we use MARL to find optimal solutions to multi-agent systems rather than controlling multiple 'agents' using a single-agent RL (SARL) algorithm?

Decomposing a large problem

- In LBF example, controlling 3 robots each with 6 actions, the joint action space becomes $6^3 = 216$.

  $\Rightarrow$ Large action space for SARL!

- We can decompose this into three independent agents, each selecting from only 6 actions.

  $\Rightarrow$ Use MARL to train separate agent policies (more tractable)

Why MARL?

Why should we use MARL to find optimal solutions to multi-agent systems rather than controlling multiple 'agents' using a single-agent RL (SARL) algorithm?

Decomposing a large problem

- In LBF example, controlling 3 robots each with 6 actions, the joint action space becomes $6^3 = 216$.

  $\Rightarrow$ Large action space for SARL!

- We can decompose this into three independent agents, each selecting from only 6 actions.

  $\Rightarrow$ Use MARL to train separate agent policies (more tractable)

Decentralized decision making

- There are many real-world scenarios where it is required for each agent to make decisions independently.
- E.g. autonomous driving is impractical for frequent long-distance data exchanges between a central agent and the vehicle.

Challenges of MARL

New **challenges** arise in MARL:

- Non-stationarity caused by multiple learning agents

Challenges of MARL

New **challenges** arise in MARL:

- Non-stationarity caused by multiple learning agents
- Optimality of policies and equilibrium selection

Challenges of MARL

New **challenges** arise in MARL:

- Non-stationarity caused by multiple learning agents
- Optimality of policies and equilibrium selection
- Multi-agent credit assignment

Challenges of MARL

New **challenges** arise in MARL:

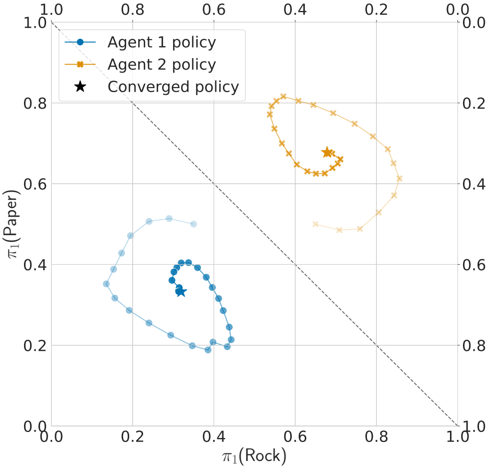
- Non-stationarity caused by multiple learning agents
- Optimality of policies and equilibrium selection
- Multi-agent credit assignment
- Scaling in number of agents

Challenges of Multi-Agent Learning

Non-stationary environment:

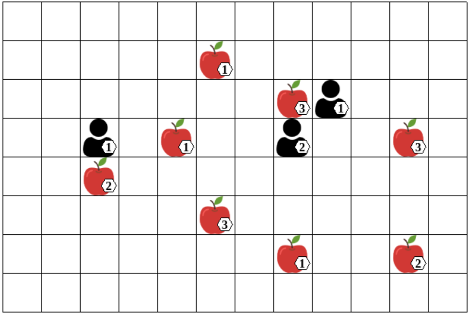
If multiple agents are learning, the environment becomes non-stationary from the perspective of individual agents

$\Rightarrow$ **Moving target:** each agent is optimizing against changing policies of other agents



Multi-Agent Credit Assignment

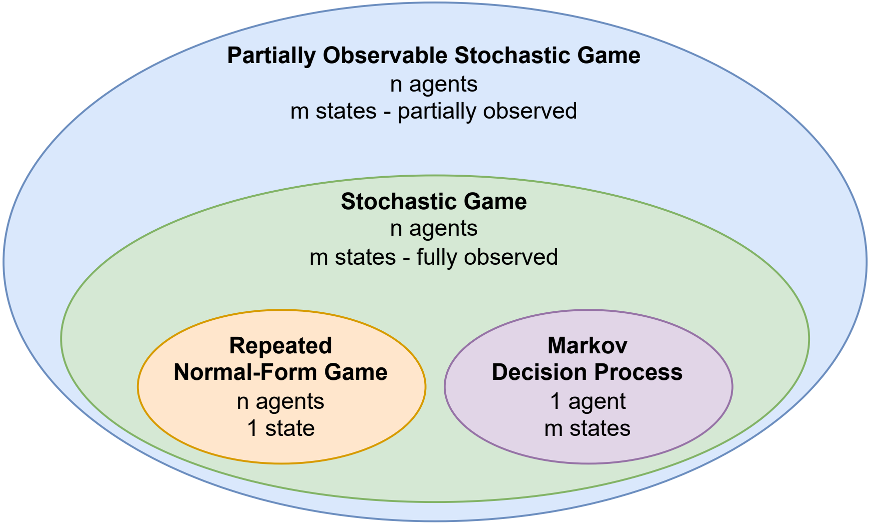
Multi-agent credit assignment: which agent's actions contributed to receved rewards?



- At time step t all agents perform *collect*, each receiving reward $+1$
- Whose actions led to the reward?
- The agent on the left did not contribute
- Learning agents must disentangle the contributions of actions!

Game models

Hierarchy of Games



Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-form games are defined as a 3 tuple $(I, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I})$:

- I is a finite set of agents $I = \{1, ..., n\}$

Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-form games are defined as a 3 tuple $(I, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I})$:

- I is a finite set of agents $I = \{1, ..., n\}$
- For each agent $i \in I$:
  - A_i is a finite set of actions
  - $\mathcal{R}_i$ is the reward function $\mathcal{R}_i : A \rightarrow \mathbb{R}$ where $A = A_1 \times ... \times A_n$ (set of **joint** actions).

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$
2. The resulting actions from all agents form a **joint action**, $a = (a_1, ..., a_n)$

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$
2. The resulting actions from all agents form a **joint action**, $a = (a_1, ..., a_n)$
3. Each agent receives a reward based on its **individual** reward function and the **joint action**, $r_i = \mathcal{R}_i(a)$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0
  i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0 i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$
- In **common-reward** games, all agents receive the same reward $(R_i = R_j; \forall i, j \in I)$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0
  i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$
- In **common-reward** games, all agents receive the same reward ($R_i = R_j; \forall i, j \in I$)
- In **general-sum** games, there are no restrictions on the relationship between reward functions.

Matrix Games

Normal-from games with **2** agents are also called **matrix games** because they can be represented using reward matrices.

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

Rock-Paper-Scissors

	A	B
A	10	0
B	0	10

Coordination Game

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Matrix Games

Normal-from games with **2** agents are also called **matrix games** because they can be represented using reward matrices.

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

Rock-Paper-Scissors
zero-sum

	A	B
A	10	0
B	0	10

Coordination Game

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Matrix Games

Normal-from games with **2** agents are also called **matrix games** because they can be represented using reward matrices.

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

Rock-Paper-Scissors
zero-sum

	A	B
A	10	0
B	0	10

Coordination Game
common-reward

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Matrix Games

Normal-from games with **2** agents are also called **matrix games** because they can be represented using reward matrices.

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

Rock-Paper-Scissors
zero-sum

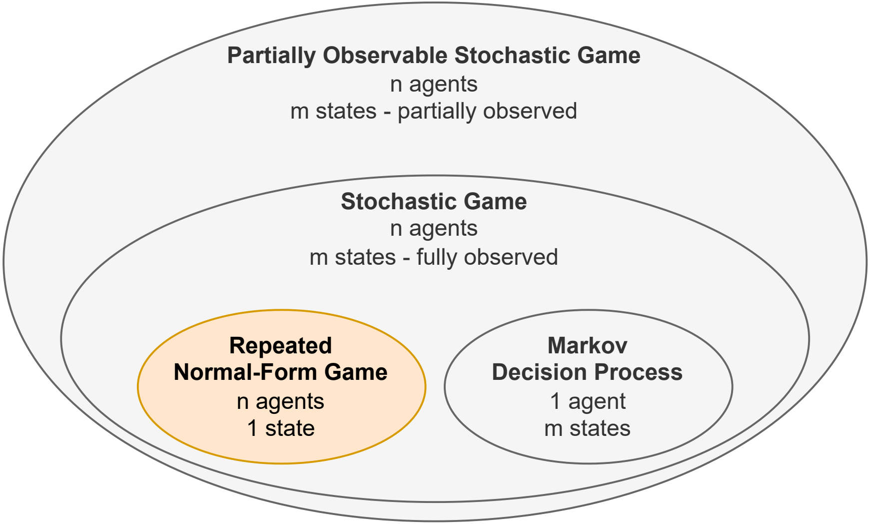
	A	B
A	10	0
B	0	10

Coordination Game
common-reward

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

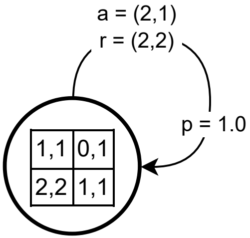
Prisoner's Dilemma
general-sum

Repeated Normal-Form Games



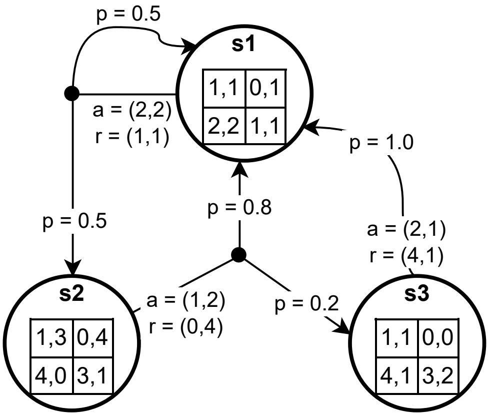
Repeated Normal-Form Games

To extend normal-form games to **sequential** multi-agent interaction, we can repeat the same game over T timesteps.



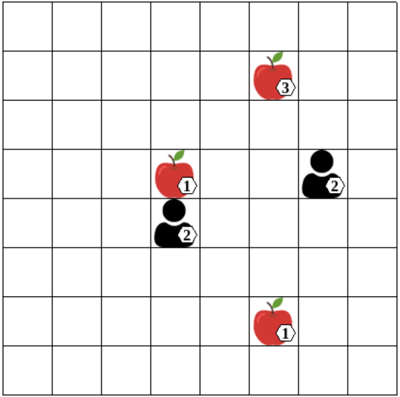
- At each time step t an agent i samples an action a_i^t
- The policy is now conditioned on a **joint-action** history $\pi_i(a_i^t|h^t)$ where $h^t = (a^o, ..., a^{t-1})$
- In special cases, h^t contains n last joint actions. E.g. in a tit-for-tat strategy (Axelrod and Hamilton 1981), the policy is conditioned on a^{t-1}

Stochastic Games



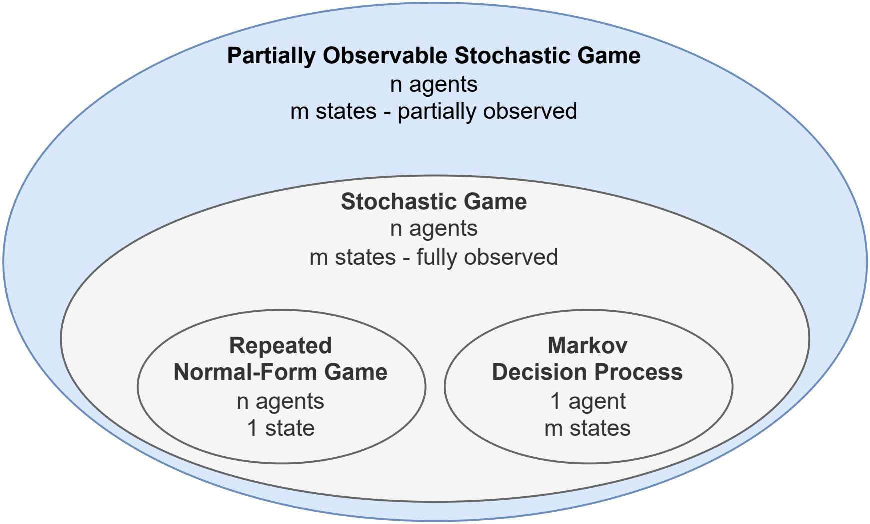
- Each **state** can be viewed as a **non-repeated normal-form game**
- Stochastic games can also be classified into: zero-sum, common-reward or general-sum
- The figure on the left shows a general-sum case

Example: Level-Based Foraging



- $s \in S$: vector of x-y positions for agents/items, binary collection flags, levels for agents/items
- $a_i \in A_i$: move in four directions, collect item, or no operation (noop)
- $\mathcal{T}$: joint actions update state, e.g., two agents collecting an item switch its flag
- $\mathcal{R}$:
  - common-reward: $+1$ reward for any item collected by any agent
  - general-sum: $+1$ reward only for agents directly involved in item collection

Partially Observable Stochastic Games (POSG)



Partially Observable Stochastic Games (POSG)

At the top of the game model hierarchy, the most **general** model is the POSG

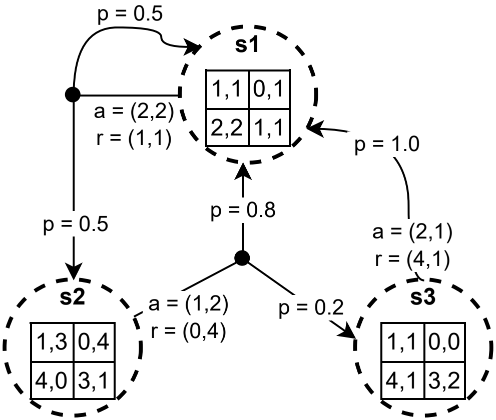
- POSGs represent complex decision processes with **incomplete information**
- Unlike in stochastic games, agents receive **observations** providing **incomplete information** about the state and agents' actions
- POSGs apply to scenarios where agents have limited sensing capabilities

  $\Rightarrow$ e.g. autonomous driving

  $\Rightarrow$ e.g. strategic games (e.g. card games) with private, hidden information

POSG Definition

POSG is defined in the same way stochastic games are, with two additions. Thus it is defined as a 8 tuple $(I, S, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I}, \mathcal{T}, \mu, \{O_i\}_{i \in I}, \{\mathcal{O}_i\}_{i \in I})$



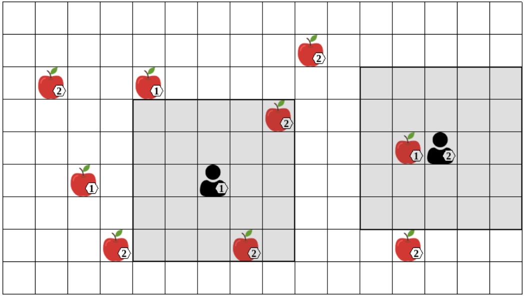
For each agent i we additionally define:

- a finite set of observation O_i
- an observation function $\{\mathcal{O}_i\}_{i \in I}$ such that $\mathcal{O}_i : A \times S \times O_i \rightarrow [0, 1]$ and $\forall a \in A, s \in S : \sum_{o_i \in O_i} \mathcal{O}_i(a, s, o_i) = 1$

The Observation Function

POSG can represent diverse observability conditions. For example:

- modeling noise by adding uncertainty in the possible observation
- to limit the visibility region of agents (see LBF example)



- Here, the agent can only see some parts of the state and joint action
- $o_i^t = (\bar{s}^t, \bar{a}^t)$ where $\bar{s}^t \subset s^t, \bar{a}^t \subset a^t$

Solution concepts for games

Solution Concepts for Games

MARL Problem	=	Game Model *e.g. normal-form game, stochastic game, POSG*	+	Solution Concept *e.g. Nash equilibrium, social welfare, ...*

What does it mean to act optimally in a multi-agent system?

- Maximizing returns of one agent is no longer enough to determine a solution
- We must consider the **joint policy** of all agents
- This is less straightforward, and there are many different solution concepts

Nash Equilibrium

Nash equilibrium solution concept applies the idea of a **mutual best response** to general-sum games with two or more agents.

- No agent i can improve its expected returns by changing its policy π_i assuming other agents policies remain fixed:

$$\forall i, \pi_i' : U_i(\pi_i', \pi_{-i}) \leq U_i(\pi)$$

- Each agent's policy is a **best response** to all other agent's policies
- John Nash (1950) proved the existence of such a solution for **general-sum non-repeated normal-from games**

Nash Equilibrium in Matrix Games

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoners Dilemma

	A	B
A	10	0
B	0	10

Coordination 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

Rock Paper Scissors

Can you identify the Nash equilibria?

Nash Equilibrium in Matrix Games

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoners Dilemma
NE at D, D (-3, -3)

	A	B
A	10	0
B	0	10

Coordination 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

Rock Paper Scissors

Nash Equilibrium in Matrix Games

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoners Dilemma

NE at D, D (-3, -3)

	A	B
A	10	0
B	0	10

Coordination Game

Two NE's at A, A (10) and B, B (10)

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

Rock Paper Scissors

Nash Equilibrium in Matrix Games

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoners Dilemma
NE at D, D (-3, -3)

	A	B
A	10	0
B	0	10

Coordination Game
Two NE's at A, A (10) and B, B (10)

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

Rock Paper Scissors
NE is to choose actions uniformly at random with expected return 0

Shortcomings of Equilibrium Solutions

Equilibrium solutions are standard solution concepts in MARL, but have limitations.

Shortcomings of Equilibrium Solutions

Equilibrium solutions are standard solution concepts in MARL, but have limitations.

- **Sub-optimality:**
    - Nash equilibria do not always maximize expected returns
    - E.g. in Prisoner's Dilemma, (D,D) is Nash but (C,C) yields higher returns

Shortcomings of Equilibrium Solutions

Equilibrium solutions are standard solution concepts in MARL, but have limitations.

- **Sub-optimality:**
  - Nash equilibria do not always maximize expected returns
  - E.g. in Prisoner's Dilemma, (D,D) is Nash but (C,C) yields higher returns
- **Non-uniqueness:**
  - There can be multiple (even infinitely many) equilibria, each with different expected returns

Shortcomings of Equilibrium Solutions

Equilibrium solutions are standard solution concepts in MARL, but have limitations.

- **Sub-optimality:**
  - Nash equilibria do not always maximize expected returns
  - E.g. in Prisoner's Dilemma, (D,D) is Nash but (C,C) yields higher returns
- **Non-uniqueness:**
  - There can be multiple (even infinitely many) equilibria, each with different expected returns
- **Incompleteness:**
  - Equilibria for sequential games don't specify actions for **off-equilibrium paths**, i.e. paths not specified by equilibrium policy $\Pr(\hat{h}|\pi) = 0$
  - If there is a temporary disturbance that leads to an off-equilibrium path, the equilibrium policy π does not specify actions to return to a **on-equilibrium** path

Refinement concepts

Pareto Optimality

To address some of these limitations, we can add additional solution requirements such as **Pareto optimality**.

A joint policy π is **Pareto-optimal** if it is not **Pareto-dominated** by any other joint policy. A joint policy π is Pareto-dominated by another policy π' if

$$\forall i : U_i(\pi') \geq U_i(\pi) \quad \text{and} \quad \exists i : U_i(\pi') > U_i(\pi).$$

Intuition

A joint policy is **Pareto-optimal** if there is no other joint policy that improves the expected return for at least one agent without reducing the expected return for any other agent.

Social Welfare and Fairness

To further constrain the space of desirable solutions, we can consider social welfare and fairness concepts.

Welfare optimality:

$$W(\pi) = \sum_{i \in I} U_i(\pi)$$

- A joint policy π is **welfare-optimal** if $\pi \in \arg\max_{\pi'} W(\pi')$

Fairness optimality:

$$F(\pi) = \prod_{i \in I} U_i(\pi), \quad U_i(\pi) > 0 \ \forall i$$

- A joint policy π is **fairness-optimal** if $\pi \in \arg\max_{\pi'} F(\pi')$

MARL learning framework

Single-Agent RL Reductions

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

Single-Agent RL Reductions

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

Central learning:

- Apply one single-agent RL algorithm to control all agents centrally
  $\Rightarrow$ A central policy is learned over the joint action space
- What is the central reward? Scaling!

Single-Agent RL Reductions

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

Central learning:

- Apply one single-agent RL algorithm to control all agents centrally

  $\Rightarrow$ A central policy is learned over the joint action space
- What is the central reward? Scaling!

Independent learning:

- Apply single-agent RL algorithms to each agent independently

  $\Rightarrow$ Agents do not explicitly consider or represent each other
- Possibly suboptimal.

MARL Challenges

Singe-Agent RL Challenges

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

MARL Challenges

Singe-Agent RL Challenges

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

Multi-Agent RL Challenges

- Non-stationarity from multiple learning agents
- Equilibrium selection
- Multi-agent credit assignment
- Scaling to many agents

Modes of Operation in MARL

Modes of operation in MARL:

Self-play:

- *Algorithm self-play:* all agents use the same learning algorithm (and parameters)
- *Policy self-play:* agent's policy is trained directly against itself

Mixed-play:

- Agents use different learning algorithms

Agent modeling

Agent Modeling & Best Response

Game theory solutions are normative: they *prescribe* how agents *should* behave

- e.g. minimax assumes worst-case opponent

Agent Modeling & Best Response

Game theory solutions are normative: they *prescribe* how agents *should* behave

- e.g. minimax assumes worst-case opponent

What if agents don't behave as prescribed by solution?

- e.g. minimax-Q was unable to exploit hand-built opponent in soccer example

Agent Modeling & Best Response

Game theory solutions are normative: they *prescribe* how agents *should* behave

- e.g. minimax assumes worst-case opponent

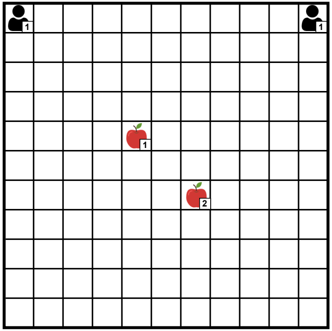
What if agents don't behave as prescribed by solution?

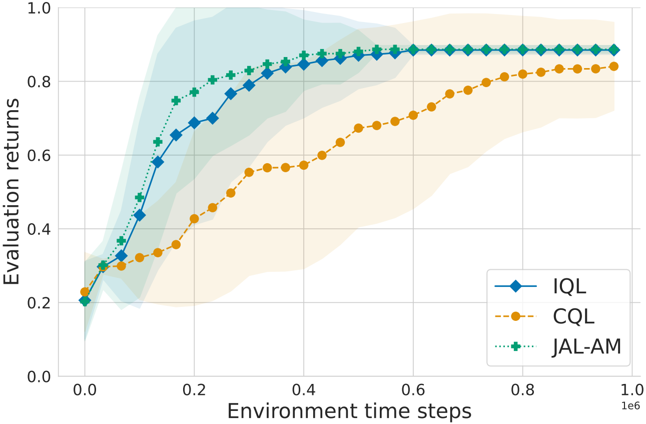
- e.g. minimax-Q was unable to exploit hand-built opponent in soccer example

Other approach: **agent modeling with best response**

- Learn models of other agents to predict their actions
- Compute optimal action (best response) against agent models

Joint Action Learnng with Agent modeling in Level-Based Foraging





Deep MARL

Challenges of Multi-Agent Reinforcement Learning

Reminder

MARL algorithms suffer from multi-agent specific challenges:

- **Non-stationarity**: exacerbated due to changing policies of all agents
- **Equilibrium selection**: how to converge to a stable equilibrium?
- **Multi-agent credit assignment**: how to attribute rewards to agents' actions?
- **Scaling to many agents**: how to efficiently scale to large numbers of agents?

Challenges of Multi-Agent Reinforcement Learning

Reminder

MARL algorithms suffer from multi-agent specific challenges:

- **Non-stationarity**: exacerbated due to changing policies of all agents
- **Equilibrium selection**: how to converge to a stable equilibrium?
- **Multi-agent credit assignment**: how to attribute rewards to agents' actions?
- **Scaling to many agents**: how to efficiently scale to large numbers of agents?

Centralised training with decentralised execution (CTDE) can help address some of these challenges.

The Multi-Agent Policy-Gradient Theorem

Solution

In MARL, the expected returns of agent i under its policy π_i depends on the policies of all other agents π_{-i} $\rightarrow$ the multi-agent policy gradient theorem defines an expectation over the policies of **all** agents (h_i: individual observation history):

The Multi-Agent Policy-Gradient Theorem

Solution

In MARL, the expected returns of agent i under its policy π_i depends on the policies of all other agents π_{-i} $\rightarrow$ the multi-agent policy gradient theorem defines an expectation over the policies of **all** agents (h_i: individual observation history):

$$\nabla_{\phi_i} J(\phi_i) \propto \mathbb{E}_{\hat{h} \sim \Pr(\hat{h}|\pi), a_i \sim \pi_i, a_{-i} \sim \pi_{-i}} \left[Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle) \nabla_{\phi_i} \log \pi_i(a_i \mid h_i = \sigma_i(\hat{h}); \phi_i) \right]$$

The Multi-Agent Policy-Gradient Theorem

Solution

In MARL, the expected returns of agent i under its policy π_i depends on the policies of all other agents π_{-i} $\rightarrow$ the multi-agent policy gradient theorem defines an expectation over the policies of **all** agents (h_i: individual observation history):

$$\nabla_{\phi_i} J(\phi_i) \propto \mathbb{E}_{\hat{h} \sim \Pr(\hat{h} | \pi), a_i \sim \pi_i, a_{-i} \sim \pi_{-i}} \left[Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle) \nabla_{\phi_i} \log \pi_i(a_i \mid h_i = \sigma_i(\hat{h}); \phi_i) \right]$$

Derive policy update rules by finding estimators for expected return $Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle)$

The Multi-Agent Policy-Gradient Theorem

Solution

In MARL, the expected returns of agent i under its policy π_i depends on the policies of all other agents π_{-i} $\rightarrow$ the multi-agent policy gradient theorem defines an expectation over the policies of **all** agents (h_i: individual observation history):

$$\nabla_{\phi_i} J(\phi_i) \propto \mathbb{E}_{\hat{h}\sim \Pr(\hat{h}|\pi), a_i \sim \pi_i, a_{-i}\sim\pi_{-i}} \left[Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle) \nabla_{\phi_i} \log \pi_i(a_i \mid h_i = \sigma_i(\hat{h}); \phi_i) \right]$$

Derive policy update rules by finding estimators for expected return $Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle)$

Independent Advantage AC (A2C) estimates $Adv(h_i, a_i) \approx Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle)$

The Multi-Agent Policy-Gradient Theorem

Solution

In MARL, the expected returns of agent i under its policy π_i depends on the policies of all other agents π_{-i} $\rightarrow$ the multi-agent policy gradient theorem defines an expectation over the policies of **all** agents (h_i: individual observation history):

$$\nabla_{\phi_i} J(\phi_i) \propto \mathbb{E}_{\hat{h} \sim \Pr(\hat{h}|\pi), a_i \sim \pi_i, a_{-i} \sim \pi_{-i}} \left[Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle) \nabla_{\phi_i} \log \pi_i(a_i \mid h_i = \sigma_i(\hat{h}); \phi_i) \right]$$

Derive policy update rules by finding estimators for expected return $Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle)$

Independent Advantage AC (A2C) estimates $Adv(h_i, a_i) \approx Q_i^{\pi}(\hat{h}, \langle a_i, a_{-i} \rangle)$

But can we do better? Perhaps by leveraging more information?

Centralized Critics

Note

In actor-critic algorithms, only the policy/actor is used during execution and the critic is used only during training $\rightarrow$ the critic can be conditioned on centralised information z without compromising decentralised execution.

Centralized Critics

Note

In actor-critic algorithms, only the policy/actor is used during execution and the critic is used only during training $\rightarrow$ the critic can be conditioned on centralised information z without compromising decentralised execution.

This might include:

- Global state s
- Joint action a
- Joint observation history h
- $\ldots$

Agent Modeling with Deep Learning

Agents Modeling – Motivation

In MARL, agents need to consider the policies of other agents to coordinate their actions.

Agents Modeling – Motivation

In MARL, agents need to consider the policies of other agents to coordinate their actions.

Approaches presented so far account for the action selection of other agents through:

- Distribution of training data is dependent on the policies of all agents
- Training centralized critics conditioned on the actions of other agents

Agents Modeling – Motivation

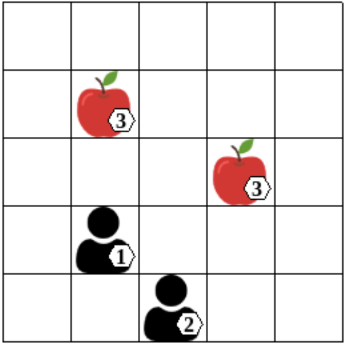
In MARL, agents need to consider the policies of other agents to coordinate their actions.

Approaches presented so far account for the action selection of other agents through:

- Distribution of training data is dependent on the policies of all agents
- Training centralized critics conditioned on the actions of other agents

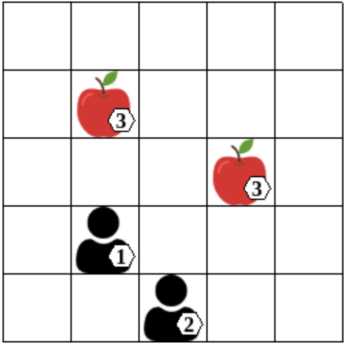
> Can we provide agents with more **explicit** information about the policies of other agents so they can learn to coordinate better, e.g. by learning best-response policies?

Joint-Action Learning with Deep Agent Models in LBF

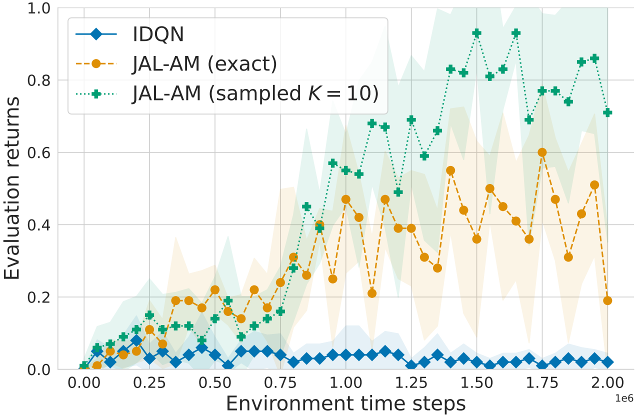


(a) Environment

Joint-Action Learning with Deep Agent Models in LBF



(a) Environment



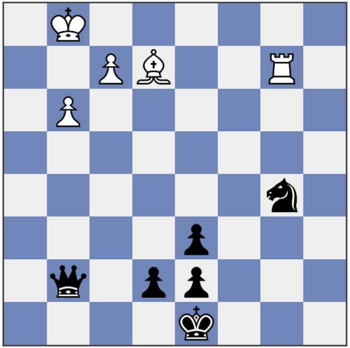
(b) Learning curve

Self-Play Monte Carlo Tree Search

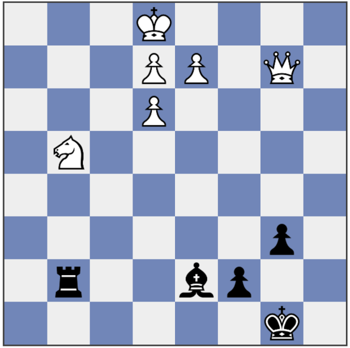
In zero-sum games with symmetrical roles and egocentric observations, agents can use the same policy to control both players

Self-Play Monte Carlo Tree Search

In zero-sum games with symmetrical roles and egocentric observations, agents can use the same policy to control both players $\rightarrow$ learn a policy in self-play



(a) Agent 1 perspective



(b) Agent 2 perspective

Population-Based Training – Self-Play for General-Sum Games

Problem

With MCTS, we focused on policy self-play in **two-agent zero-sum** games. Can we extend the idea of self-play to **general-sum** games with **more than two agents**?

Population-Based Training – Self-Play for General-Sum Games

Problem

With MCTS, we focused on policy self-play in **two-agent zero-sum** games. Can we extend the idea of self-play to **general-sum** games with **more than two agents**?

Population-based training is a generalisation of self-play to general-sum games:

- Maintain a **population of policies** representing possible strategies of the agent
- Evolve populations so they become more effective against the populations of other agents
- We denote the population of policies for agent i at generation k as Π_i^k.

This lecture was based on

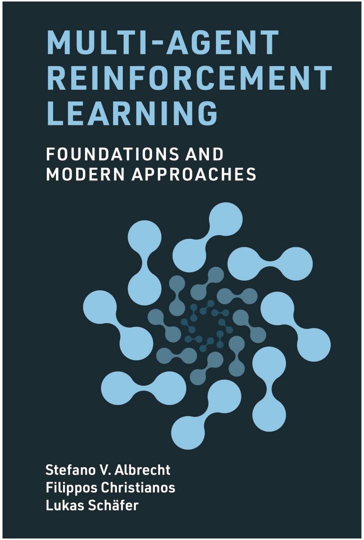
Multi-Agent Reinforcement Learning: Foundations and Modern Approaches

by Stefano V. Albrecht, Filippos Christianos and Lukas Schäfer

MIT Press, 2024

Download book, slides, and code at:

www.marl-book.com



Outlook: Whither RL?

Current challenges in RL

- DRL, POMDP, MARL, MORL, IRL, ...
- Sample complexity, exploration, stability, robustness, generalisation, reproducibility
- Problem characteristics, Meta-RL, transfer learning, reward shaping, RL + LLM
- Data-based (off-line) RL, sim-to-real gap, latency, learning from many policies
- Biological RL, population-based methods, naturalistic RL (Wise e.a., 2024)
- Safe RL, human-in-the-loop RL, grounded RL, interdisciplinary problems
- Applications: Industry size of RL: £100B (2025) growth rate 65% (rough estimates!).