Multi-agent Reinforcement Learning

Mattia Pellegrino, Ph.D Fellow
mattia.pellegrino@unipr.it

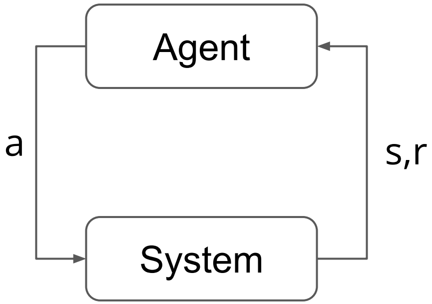
OVERVIEW

- Recap: Markov Models
- Multi-agent Reinforcement Learning (MARL)

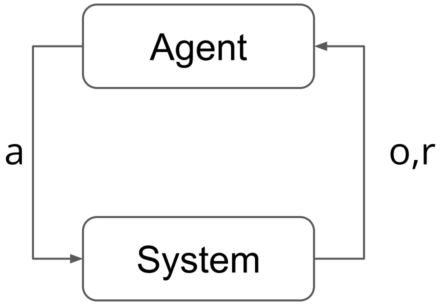
RECAP – MARKOV MODEL

	No agents	Single Agent	Multiple Agents
State known	Markov Chain	Markov Decision Process (MDP)	Markov Game (aka Stochastic Game)
State Observed Indirectly	Hidden Markov Model (HMM)	Partially Observable Markov Decision Process (POMDP)	Partially-Observable Stochastic Game (POSG)

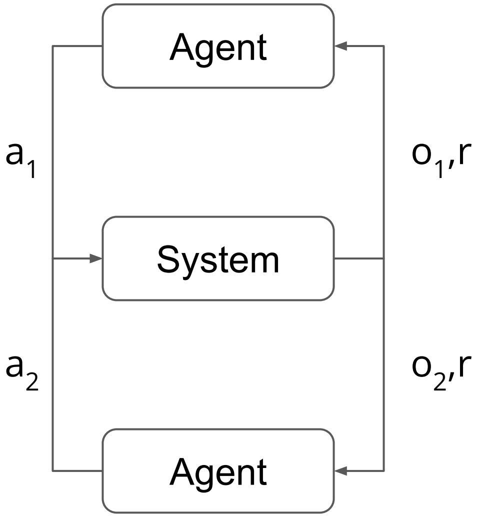
RECAP – MARKOV MODEL



MDP



POMDP



Dec-POMDP

MARL: WHAT IS MARL?

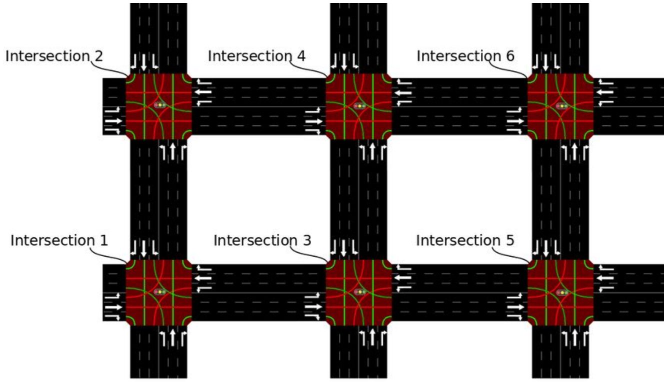
- Form of Reinforcement Learning (RL)
  - Agent(s) learn to take actions that maximize a reward derived from the environment
- Includes multiple independent actors (agents)
  - Each agents actions may change the environment
  - Changes to the environment could affect reward for all agents
- Agents may interact to maximize their reward
  - Intentional changes to the environment
  - Direct agent-to-agent communication
  - Cooperation vs competition

MARL: APPLICATIONS – DOTA 2

- Dota 2 AI agents are trained to coordinate with each other to compete against humans.
- Each of the five AI players is implemented as a separate neural network policy and trained together with large-scale PPO (Proximal Policy Optimization).

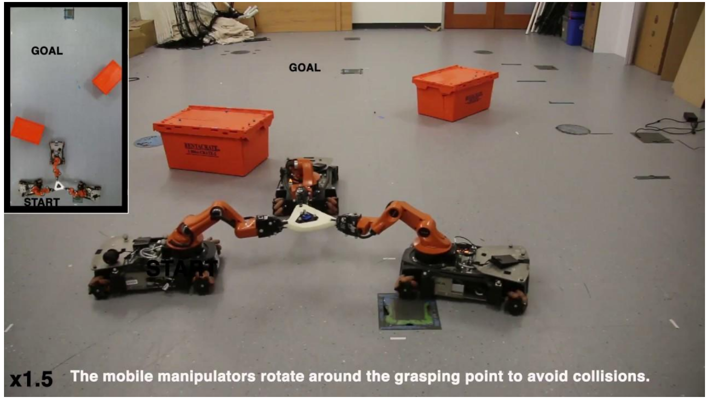
MARL: APPLICATIONS – TRAFFIC SIGNAL CONTROL

- MARL can be applied to optimize traffic signal control in urban areas. By treating each intersection as an agent.
- MARL algorithms can learn to dynamically adjust signal timings based on traffic flow patterns, congestion levels, and overall system efficiency.
- This approach can lead to reduced travel times, improved traffic flow, and reduced congestion.

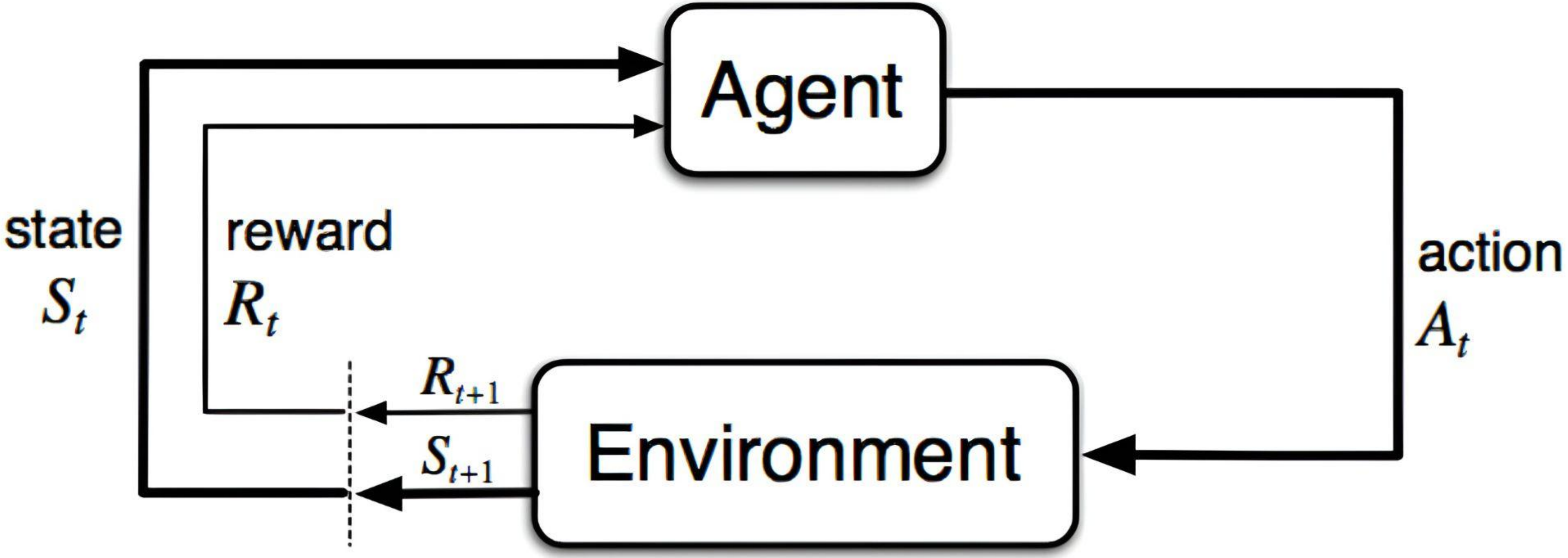


MARL: APPLICATIONS - MULTI ROBOT SYSTEMS

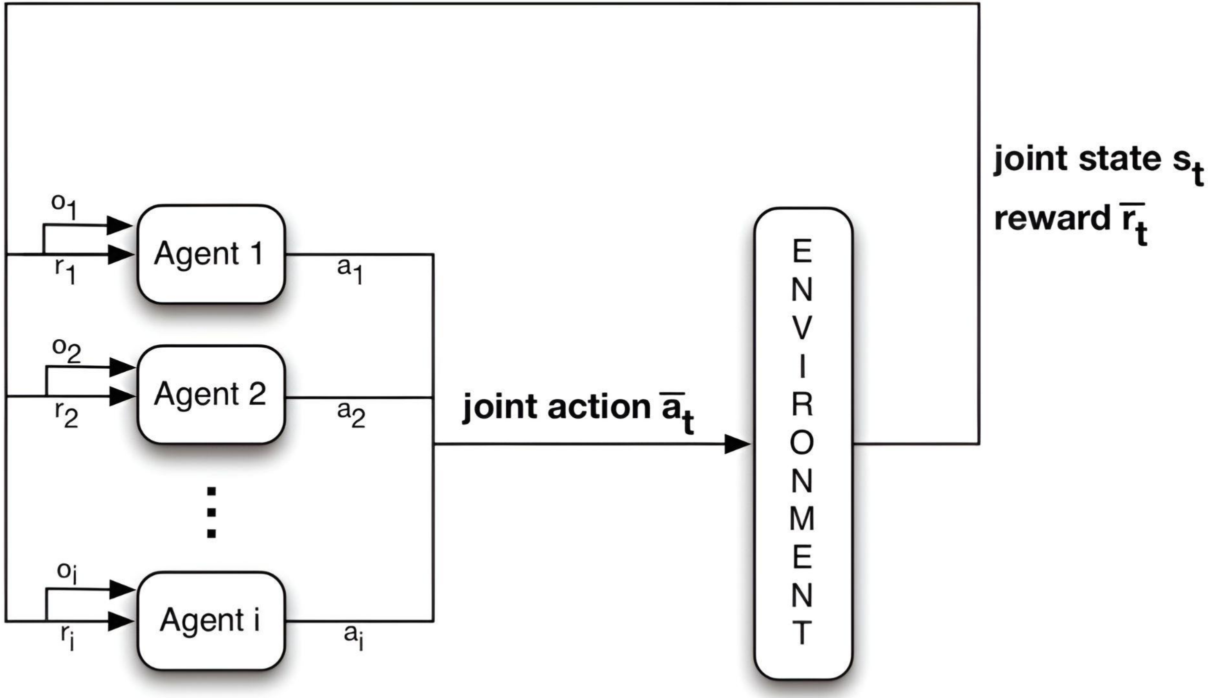
- MARL is widely used in the field of robotics to enable multiple robots to collaborate and perform complex tasks.
- For example, in warehouse automation, robots can learn to coordinate their actions to efficiently pick and transport items, optimize routing, and avoid collisions.
- MARL algorithms enable the robots to adapt and learn from each other's experiences, improving overall system performance.



MARKOV DECISION PROCESS (MDP)



MULTI-AGENT REINFORCEMENT LEARNING



Source: Nowe, Vrancx & De Hauwere 2012

MARL: PROPERTIES

- **Centralized**
  - One brain / algorithm deployed across many agents
- **Decentralized**
  - All agents learn individually
  - Communication limitations defined by environment

- **Prescriptive**
  - Suggests how agents should behave
- **Descriptive**
  - Forecast how agent will behave

MARL: PROPERTIES

- **Cooperative**: Agents cooperate to achieve a goal
  - Shared team reward
- **Competitive**: Agents compete against each other
  - Zero-sum games
  - Individual opposing rewards
- **Neither**: Agents maximize their utility which may require cooperating and/or competing
  - General-sum games

MARL: PROPERTIES

- **Numbers of agents**
  - One (single-agent)
  - Two (very common)
  - Finite
  - Infinite

MULTIAGENT MODELS

- Normal-form game
- Repeated game
- Stochastic game

NORMAL-FORM "ONE-SHOT" GAME

- Normal-form game consists of:
  - Finite set of agents $i \in \mathcal{N} = \{1, \cdots, n\}$
  - Each agent i has a set of actions $A_i \in \{a_1, a_2, \cdots\}$
  - Set of joint actions $A = a_1 \times a_2 \times \cdots \times a_n$
  - Rewards function $r_i : A \to \mathbb{R}$, where $A = A_1 \times \cdots \times A_n$

- Each agent i selects policy $\pi_i : A_i \to [0, 1]$, takes action $a_i \in A_i$ with probability $\pi_i(a_i)$, and receives reward $r_i(a_1, \cdots, a_n)$. Given policy profile $(\pi_1, \cdots, \pi_n)$, expected reward to i is

$$r(\pi_1, \cdots, \pi_n) = \sum_{a \in A} \pi_1(a_1) * \cdots \pi_n(a_n) * r_i(a)$$

NORMAL-FORM: ROCK-PAPER-SCISSORS

- Two players, three actions
- Rock beats Scissors beats Paper beats Rock

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

REPEATED GAME

- Normal-form game is single interaction. No experience
  - The agent does not considering the consequences
- Experience comes from repeated interactions

- **1** - Initialization: Agents are initialized with some initial strategy
- **2** - Interaction: Agents interacts with each others
- **3** - Observation and Learning: Agents observe the actions and the rewards gained (Q-Learning, SARSA, ecc ecc)
- **4** - Strategy Update: Agent update their strategy based on learning algorithm
- **5** - Repeat: Step 2-4 for a predefined number of interactions

STOCHASTIC GAME

- Action transitions and rewards are random

- **1** - Initialization: Agents are initialized with some initial strategy
- **2** - Interaction: Agents interacts with each others
- **3** - The environment introduces some random elements
- **4** - Observation and Learning: Agents observe the actions and the rewards gained (Q-Learning, SARSA, ecc ecc)
- **5** - Strategy Update: Agent update their strategy based on learning algorithm
- **6** - Repeat: Step 2-5 for a predefined number of interactions

MULTI-AGENTS LEARNING SYSTEM

- **Sharing experience**
  - via communication, teaching, imitation
- **Parallel computation**
  - due to decentralized task structure
- **Robustness**
  - redundancy, having multiple agents to accomplish a task

MULTI-AGENTS LEARNING SYSTEM CHALLENGES

- **Curse of dimensionality**
  - Exponential growth in computational complexity from increase in state and action dimensions. Also a challenge for single-agent problems.
- **Specifying a good (learning) objective**
  - Agent returns are correlated and cannot be maximized independently.
- **The system in which to learn is a moving target**
  - As some agents learn, the system which contains these agents changes, and so may the best policy. Also called a system with non-stationary or time-dependent dynamics.
- **Need for coordination**
  - Agent actions affect other agents and could confuse other agents (or herself) if not careful. Also called destabilizing training.

FIRST MARL ALGORITHM

- **Minimax-Q (Littman -94)**
  - Q-values are over joint actions: Q(s,a,o)
    - s = state
    - a = action
    - o = actions of the opponent

maximizes its minimum guaranteed reward

Minimize the reward of the opponent

$$Q(s,a,o) = (1-\alpha)Q(s,a,o) + \alpha(r + \gamma V(s'))$$

$$V(s) = \max_{\pi_s} \min_{o} \sum_{a} Q(s,a,o)\pi_s(a)$$

MARL FORMULATION

- The agents choose actions according to their policies.
- For agent j, the corresponding policy is defined as $\pi^j : S \rightarrow \Omega(A^j)$, where $\Omega(A^j)$ is the collection of probability distributions over agent j's action space A_j
- Let $\pi = [\pi^1, \cdots, \pi^N]$ - is the joint policy of all agents, then

$$v^j_{\boldsymbol{\pi}}(s) = v^j(s; \boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\boldsymbol{\pi},p}[r^j_t | s_0 = s, \boldsymbol{\pi}]$$

- Q-function such that the Q-function $Q^j_\pi : S \times A^1 \times \cdots \times A^N \rightarrow R$ of agent j under the joint policy π:

$$Q^j_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s' \sim p}[v^j_{\boldsymbol{\pi}}(s')]$$

NASH Q-LEARNING

- In MARL, the objective of each agent is to learn an optimal policy to maximize its value function
- Optimizing the v^j_π for agent j depends on the joint policy **π** of all agents
- A Nash equilibrium is a joint policy π such that no player has incentive to deviate unilaterally. It is represented by a particular joint policy

$$\boldsymbol{\pi}_* = [\pi_*^1, \cdots, \pi_*^N]$$

- such that for all s ∈ S, j ∈ {1, · · · , N} it satisfies:

$$v^j(s; \boldsymbol{\pi}_*) = v^j(s; \pi_*^j, \boldsymbol{\pi}_*^{-j}) \geq v^j(s; \pi^j, \boldsymbol{\pi}_*^{-j})$$

- Here $\boldsymbol{\pi}^{-j}_*$ is the joint policy of all agents except j as

$$\boldsymbol{\pi}_*^{-j} = [\pi_*^1, \cdots, \pi_*^{j-1}, \pi_*^{j+1}, \cdots, \pi_*^N]$$

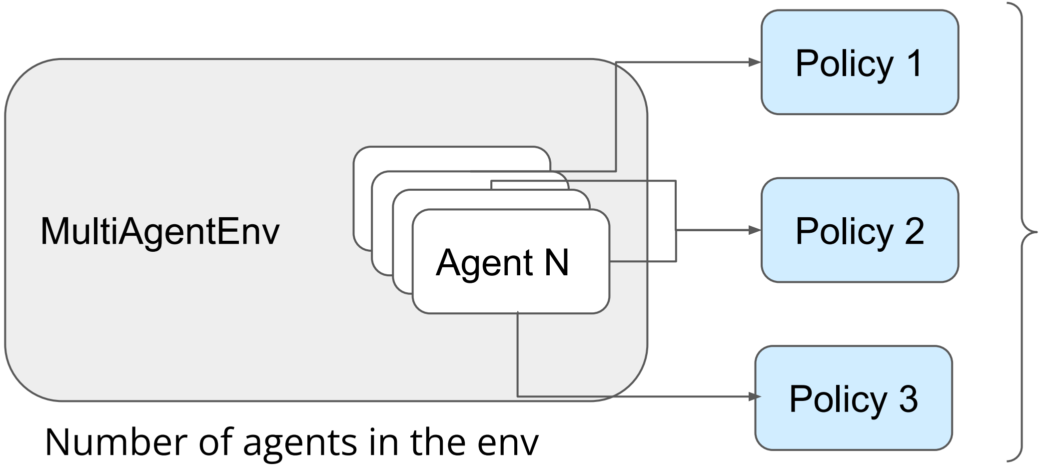
NASH Q-LEARNING

- In a Nash equilibrium, each agent acts with the best response π^j_* to others, provided that all other agents follow the policy π_*^{-j}
- For a N-agent stochastic game, there is at least one Nash equilibrium with stationary policies, assuming players are rational
- Given Nash policy π_*, the Nash value function

$$\begin{aligned} \boldsymbol{v}^{\text{Nash}} &= [v^1_{\boldsymbol{\pi}_*}(s), \cdots, v^N_{\boldsymbol{\pi}_*}(s)] \\ \boldsymbol{Q}(s, \boldsymbol{a})^{\text{Nash}} &= \mathbb{E}_{s' \sim p}[\boldsymbol{r}(s, \boldsymbol{a}) + \gamma \boldsymbol{v}^{\text{Nash}}(s')] \end{aligned}$$

where $\boldsymbol{r}(s, \boldsymbol{a}) = [r^1(s, \boldsymbol{a}), \cdots, r^N(s, \boldsymbol{a})]$

MARL POLICIES

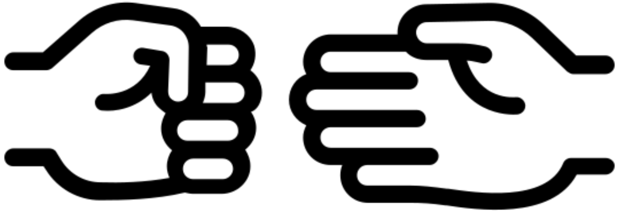


Number of agents in the env can vary over time. There may be hundreds or more agents.

Number of distinct policies is fixed. There may be up to hundreds of policies

(constrained by memory required to sore weights)

EXERCISE - ROCK SCISSORS PAPER



AGENT 1				
		Rock	Paper	Scissors
AGENT 2	Rock	0,0	-1,1	1,-1
	Paper	1,-1	0,0	-1,1
	Scissors	-1,1	1,-1	0,0

- Modify the source code (available at: https://github.com/sowide/reinforcement_learning_course [30-05_exercise folder]) and add the required comments to explain its behavior
- Deliver the modified source code: https://shorturl.at/dWX89 [name_surname.zip]