Multi-Agent (Deep) Reinforcement Learning

Aida Afshar Mohammadian
Aeirya Mohammadi

ML Course - Fall 1401
Dr. Rezazadegan

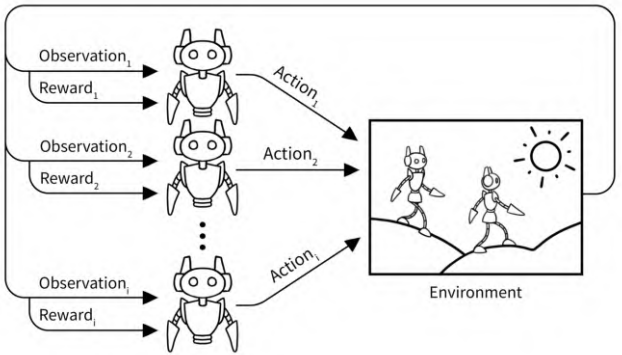
Overview

- Introduction
  - Formal + Informal Definition of MARL
- Multi-Agent Training Architectures
  - Distributed/Centralized + Training/Execution
- Reward Structure
  - Emergent patterns of agent behavior
  - Hide and Seek
- Petting Zoo

Introduction

A **multi-agent system** describes multiple distributed entities, so-called **agents**, which take decisions autonomously and interact within a shared environment. These agents require the ability to adapt and learn over time by themselves.

The most common framework to address learning in an interactive environment is **Reinforcement Learning**, which describes the change of behavior through a trial-and-error approach.

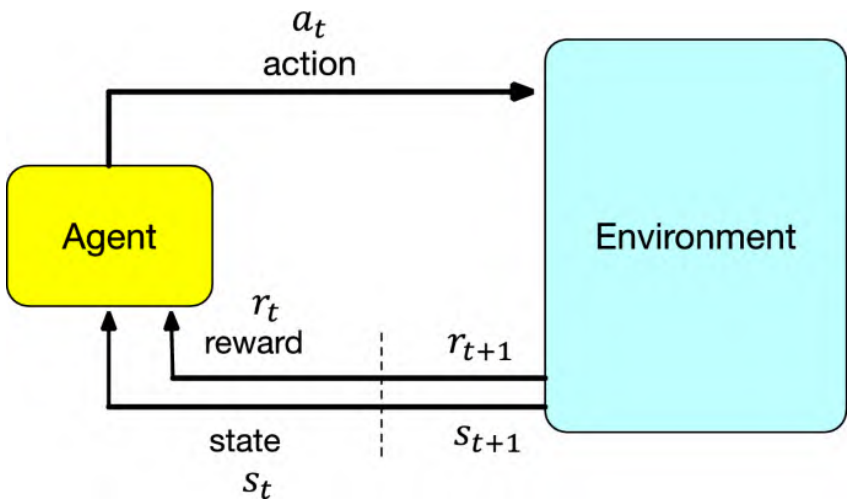


Single-Agent RL

Def 1 A Markov decision process (**MDP**) is formalized by the tuple (S, A, P, *R*, γ) where S and A are the state and action space, respectively, P : S × A → [0,1] is the transition function describing the probability of a state transition, R : S × A × S → $\mathbb{R}$ is the reward function providing an immediate feedback to the agent, and $\gamma \in$ [0, 1) describes the discount factor.

$$\bar{R} = E_{s_0 \sim \rho_0, s_{t+1} \sim P, a_t \sim \pi} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1}) \right]$$

The agent's goal is to act in such a way as to maximize the expected performance, regarding the policy π, over a predefined horizon T.



Multi-Agent RL

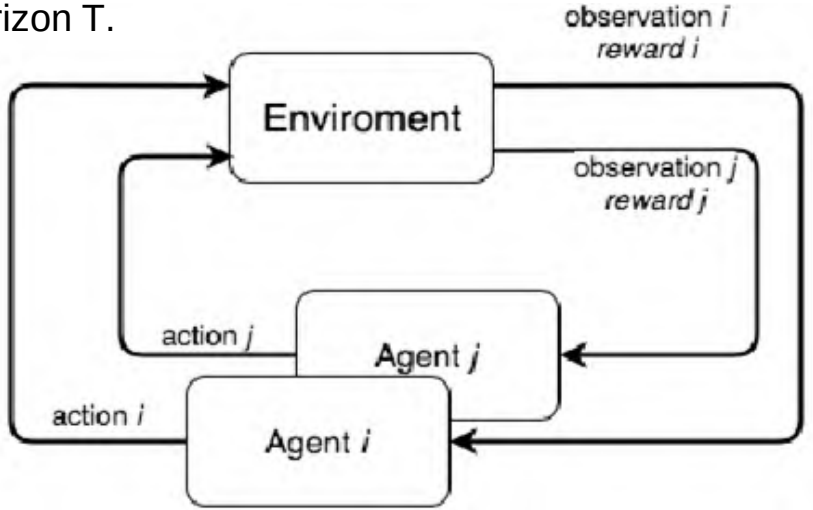
Def 2 The Markov Game (**MG**) is an extension to the MDP and is formalized by the tuple (N, S, {A*i*}, P, {*Ri*}, γ), where N = {1, ... , *N*} denotes the set of *N* > 1 interacting agents and S is the set of states observed by all agents. The joint action space is denoted by A = $A_1 \times \cdots \times A_N$ which is the collection of individual action spaces from agents $i \in N$. The transition probability function P : $S \times A \rightarrow [0,1]$ describes the chance of a state transition. Each agent owns an associated reward function $R_i : S \times A \times S \rightarrow \mathbb{R}$ that provides an immediate feedback signal. Finally, $\gamma \in [0, 1)$ describes the discount factor.

$$\bar{R}_i = E_{s_0 \sim \rho_0, s_{t+1} \sim P, a_t \sim \pi} \left[\sum_{t=0}^{T} \gamma^t R_i(s_t, a_t, s_{t+1}) \right]$$

$$a_t = (a_1, a_2, .., a_N)$$

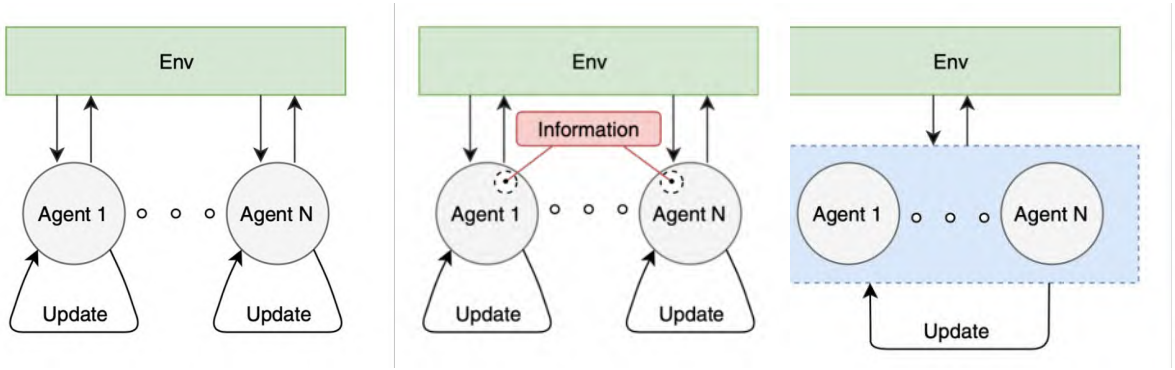
$$\pi = \pi_1 \times \pi_2 \times \ldots \times \pi_N$$

At stage t, each agent $i \in N$ selects and executes an action depending on the individual policy $\pi i : S \rightarrow P(Ai)$. The system evolves from state x under the joint action at with respect to the transition probability function P to the next state $St+1$ while each agent receives Ri as immediate feedback to the state transition. Akin to the single-agent problem, the aim of each agent is to change its policy in such a way as to optimize the received rewards on a predefined horizon T.



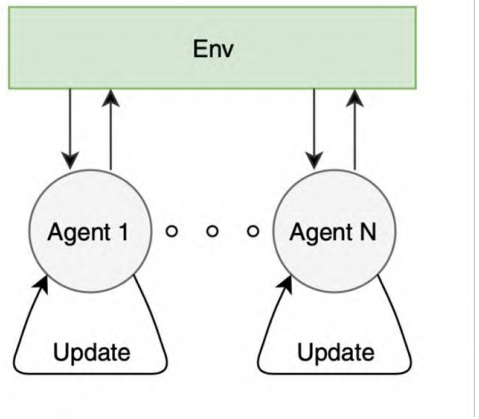
Multi-Agent Training Architectures

- Categorized based on
  - **Mutual information**
  - Agents dependency



Decentralized Training

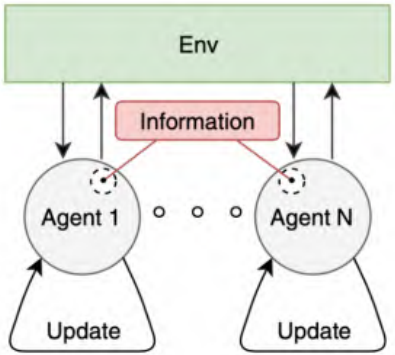
Agents determine actions according to their individual policy for decentralized execution.



Decentralized Training + Shared Information

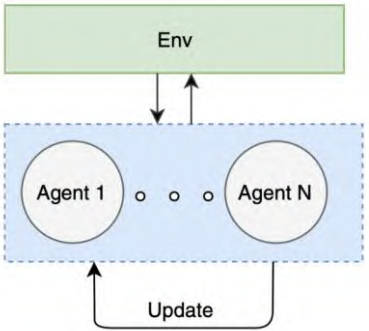
Each agent holds an individual policy. During training, agents are endowed with additional information.

- By sharing mutual information, the training process can be eased and the learning speed can become superior when matched against independently trained agents.
- The state-of-the-art practice for MARL.



Centralized Training

Centralized execution describes that agents are guided from a centralized unit, which computes the joint actions for all agents.



Reward Structure

- Emergent patterns of agents behavior

The dynamic interaction between multiple decision-makers, which simultaneously affects the state of the environment, can cause the emergence of specific behavioral patterns. An obvious way to influence the development of agent behavior is through the designed reward structure.

  - Cooperation
  - Competition
  - Mixed

Hide and Seek Game

Setup:

- 2 blue agents
- 2 red agents
- 2 boxes
- A ramp

Hide and Seek Game

Simple game, simple rules. The agents can:

- Move and rotate themselves
- Grab objects
- See objects in their sight
- Sense distance
- Lock and unlock objects

Link to website

Gym

The Gym interface is simple, pythonic, and capable of representing general RL problems.

```
import gym
env = gym.make('CartPole-v0')
observation = env.reset()
for _ in range(1000):
    action = policy(observation)
    observation, reward, done, info = env.step(action)
```

action space = { 0, 1 }

observation space = { cart position & velocity, pole angle & angular velocity }

Petting Zoo

PettingZoo is a Python library for conducting research in multi-agent reinforcement learning, akin to a multi-agent version of Gymnasium.

You can see more at here.

The Agent Environment Cycle Games Model

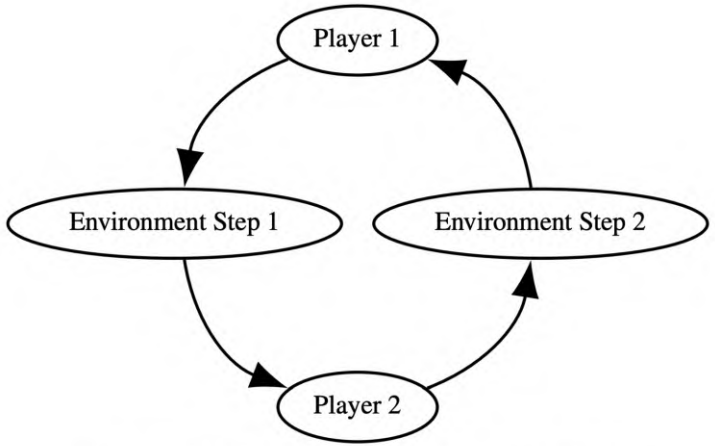
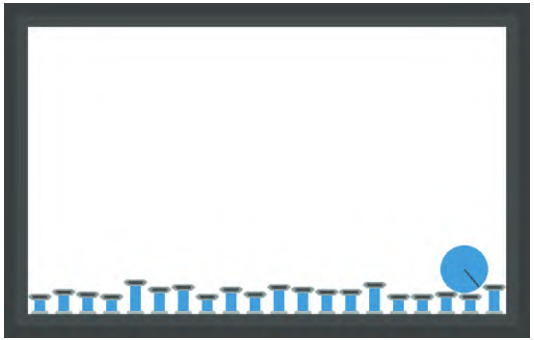


Figure 5: The AEC diagram of Chess

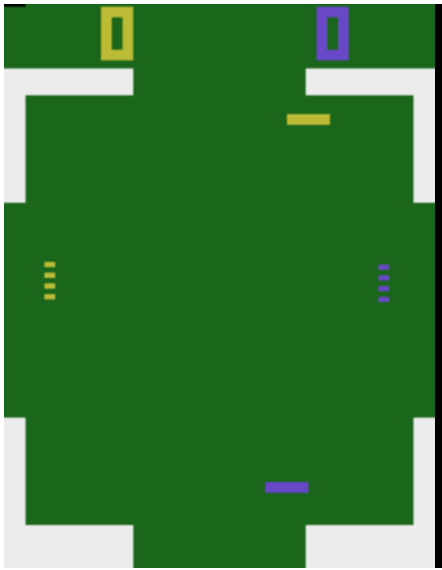
An example of the basic usage of PettingZoo

```
from pettingzoo.butterfly import pistonball_v0
env = pistonball_v0.env()
env.reset()
for agent in env.agent_iter(1000):
    env.render()
    observation, reward, done, info = env.last()
    action = policy(observation, agent)
    env.step(action)
env.close()
```



Showcase

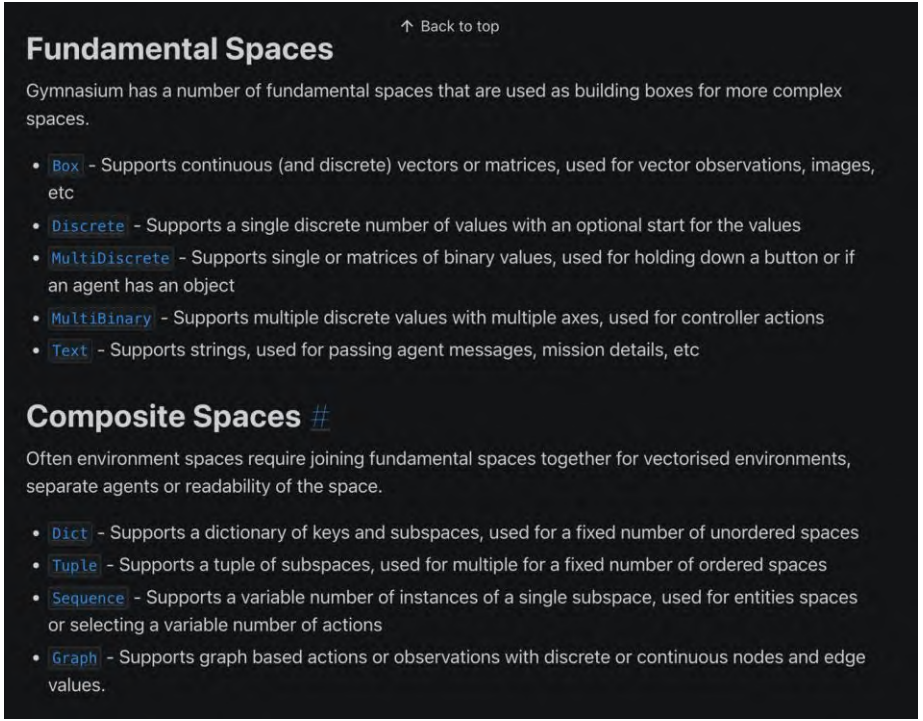
Multiple agents playing implemented petting zoo environments:
(if image is static, click to play)



Quadrapong

Knights archers zombies

Gym Spaces Data Structure



↑ Back to top

Fundamental Spaces

Gymnasium has a number of fundamental spaces that are used as building boxes for more complex spaces.

- `Box` - Supports continuous (and discrete) vectors or matrices, used for vector observations, images, etc
- `Discrete` - Supports a single discrete number of values with an optional start for the values
- `MultiDiscrete` - Supports single or matrices of binary values, used for holding down a button or if an agent has an object
- `MultiBinary` - Supports multiple discrete values with multiple axes, used for controller actions
- `Text` - Supports strings, used for passing agent messages, mission details, etc

Composite Spaces

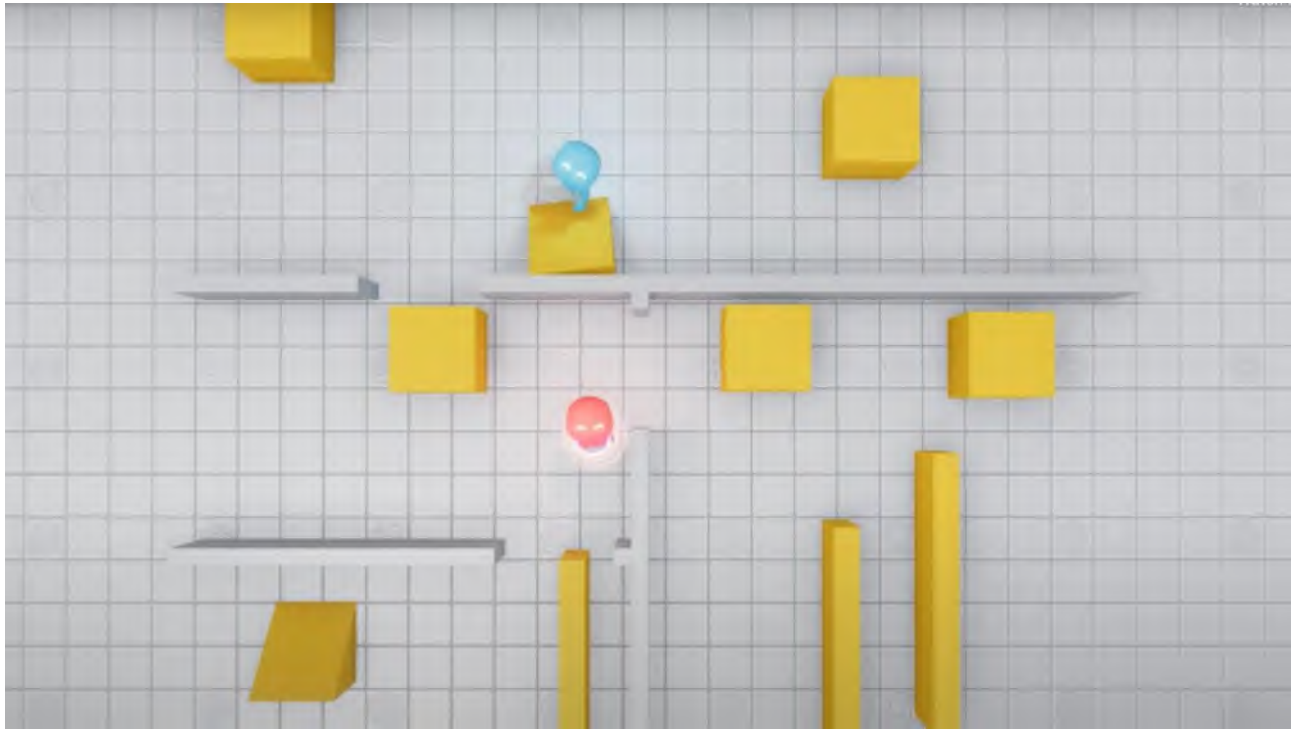
Often environment spaces require joining fundamental spaces together for vectorised environments, separate agents or readability of the space.

- `Dict` - Supports a dictionary of keys and subspaces, used for a fixed number of unordered spaces
- `Tuple` - Supports a tuple of subspaces, used for multiple for a fixed number of ordered spaces
- `Sequence` - Supports a variable number of instances of a single subspace, used for entities spaces or selecting a variable number of actions
- `Graph` - Supports graph based actions or observations with discrete or continuous nodes and edge values.

Let’s see some code!

@googlecolab: Piston - Petting Zoo Sample

@github: repository/code/cartpole_qlearning.py



Thank you for your attention!

References

Main reference: Multi-agent deep reinforcement learning: a survey

PettingZoo: PettingZoo: A Standard API for Multi-Agent Reinforcement Learning

https://pettingzoo.farama.org

https://github.com/Farama-Foundation/PettingZoo

Gym: https://www.gymlibrary.dev

https://github.com/openai/gym

Gymnasium (updated version of gym): https://gymnasium.farama.org

Hide & Seek Game: Emergent Tool Use from Multi-Agent Interaction

—

Contact us: afsharaidam@gmail.com aeirya@gmail.com