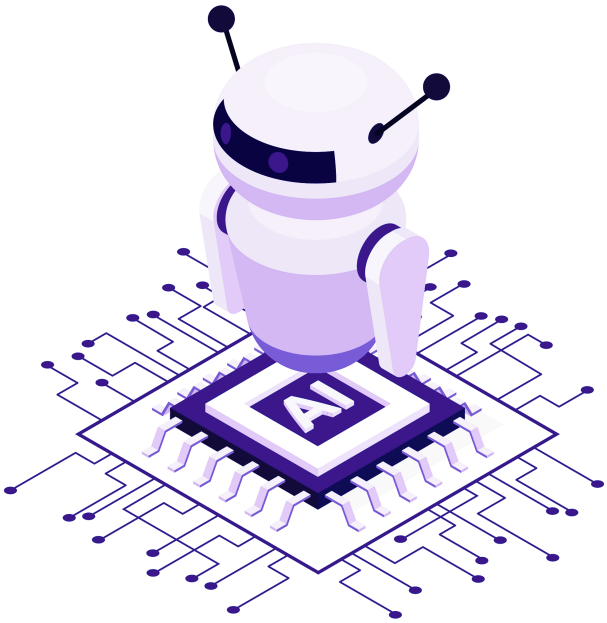

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 13:

Multi-Agent RL

Designed By:

Behnia Soleymani
rl.nerd@outlook.com

Faezeh Sadeghi
fz.saadeghi@gmail.com

RIML

Spring 2025

Preface

Welcome to the homework!

In this assignment, you'll explore the exciting intersection of multi-agent systems, where multiple intelligent agents interact, compete, and cooperate. Unlike single-agent systems, where one agent learns to interact with a fixed environment, MARL deals with multiple learning agents, each interacting not just with the environment, but also with each other. This makes the learning process richer, more complex, and often more realistic.

First, you'll tackle foundational problems from **Game Theory**, analyzing strategic decision-making through concepts like Nash equilibria and regret minimization.

Next, you'll implement modern multi-agent learning algorithms such as **MADDPG** (Multi-Agent Deep Deterministic Policy Gradient) and **Independent DDPG**. These approaches combine centralized training and decentralized execution, enabling agents to effectively coordinate their actions.

By blending theoretical insights and practical implementations, you'll gain hands-on experience with how intelligent agents learn and strategize in complex environments.

Grading

The grading will be based on the following criteria, with a total of 110 points:

Task	**Points**
Task 1	50
Task 2	50
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1	5
Bonus 2	5

Submission

The deadline for this homework is 1404/06/20 (Sep 11th 2025) at 11:59 PM. Please submit your work by following the instructions below:

- Place your solution alongside the Jupyter notebook(s).
  - Your written solution must be a single PDF file named `HW13_Solution.pdf`.
  - If there is more than one Jupyter notebook, put them in a folder named `Notebooks`.
- Zip all the files together with the following naming format:

  `DRL_HW13_[StudentNumber]_[FullName].zip`

  - Replace `[FullName]` and `[StudentNumber]` with your full name and student number, respectively. Your `[FullName]` must be in CamelCase with no spaces.
- Submit the zip file through Quera in the appropriate section.
- We provided this LaTeX template for writing your homework solution. There is a 5-point bonus for writing your solution in LaTeX using this template and including your LaTeX source code in your submission, named `HW13_Solution.zip`.
- If you have any questions about this homework, please ask them in the Homework section of our Telegram Group.
- If you are using any references to write your answers, consulting anyone, or using AI, please mention them in the appropriate section. In general, you must adhere to all the rules mentioned here and here by registering for this course.

Keep up the great work and best of luck with your submission!

Contents

1 Part 1: Game Theory Problems

Game theory provides mathematical frameworks for modeling strategic interactions among rational decision-makers, known as agents. In multi-agent systems, each agent aims to maximize its own payoff or reward, considering the actions of other agents. Game theory enables us to analyze and predict stable outcomes, known as equilibria, where no agent can benefit by changing their strategy unilaterally. Understanding these equilibria is essential for designing agents capable of rational and strategic decision-making in interactive scenarios.

Now, In the first notebook (`HW_13_first_part.ipynb`), four problems related to foundational concepts in Game Theory have been defined. Each problem includes detailed explanations and instructions for implementation, particularly focusing on algorithms such as Fictitious Play and Regret Minimization. You will analyze strategic interactions between agents, implement theoretical concepts practically, and observe convergence behaviors within classic game settings. Complete each problem as instructed in the notebook.

2 Part 2: Implementing MADDPG/IDDPG

In this part of the assignment, you will implement the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [2] and its independent variant (IDDPG). These methods are core algorithms used in multi-agent reinforcement learning to train agents to coordinate their actions effectively within a shared environment. You will use the TorchRL library, a powerful PyTorch extension designed for developing advanced RL algorithms efficiently. Detailed instructions, along with clearly marked `TODO` sections, are provided in the assignment script. Follow these instructions to complete the implementation (Read more).

2.1 VMAS Environment and Navigation Scenario

In this assignment, you will use the Vectorized Multi-Agent Simulator (VMAS), a customizable and efficient framework for testing and benchmarking multi-agent reinforcement learning algorithms. VMAS allows parallel simulation of multiple agents interacting within a shared environment.

Specifically, you will work with the **navigation scenario**, where multiple agents must independently navigate to assigned target positions while avoiding collisions and optimizing cooperative behavior. Each agent observes its own position, velocity, and goal position. Agents receive rewards based on how efficiently they reach their targets, with penalties for collisions or suboptimal paths.

This scenario provides a practical setting for exploring and evaluating the effectiveness of your MADDPG and IDDPG implementations in a cooperative multi-agent environment.

2.2 TorchRL

TorchRL is a powerful and flexible PyTorch-based library designed specifically for reinforcement learning (RL) research and development. It provides modular and intuitive tools, such as efficient replay buffers, built-in RL loss functions, and data collectors, enabling streamlined implementation and experimentation with both single-agent and multi-agent reinforcement learning algorithms. TorchRL's design emphasizes ease of use, scalability, and integration with PyTorch's ecosystem.

For detailed documentation and examples, please refer to the official TorchRL documentation:

`https://docs.pytorch.org/rl/stable/index.html`

and specifically

`https://docs.pytorch.org/rl/stable/tutorials/multiagent_competitive_ddpg.html`

2.3 Questions

1. In our training loop, the `DDPGLoss` module utilizes `target_policies` to estimate the value of the next state. Explain clearly why employing these slowly-updating target networks, rather than the main policy networks (which change rapidly), is essential for ensuring the stability of the DDPG algorithm. (Hint: Consider what might happen if the critic tried to optimize toward a continuously moving target.)
2. (bonus) Consider the training plot shown in Figure 1, which resulted from modifying a single scalar hyper-parameter in the training script.
   (a) Describe the issue with the learning process depicted in the plot.

(b) Identify which hyper-parameter you believe was changed, and explain the role of this parameter within the MADDPG algorithm.

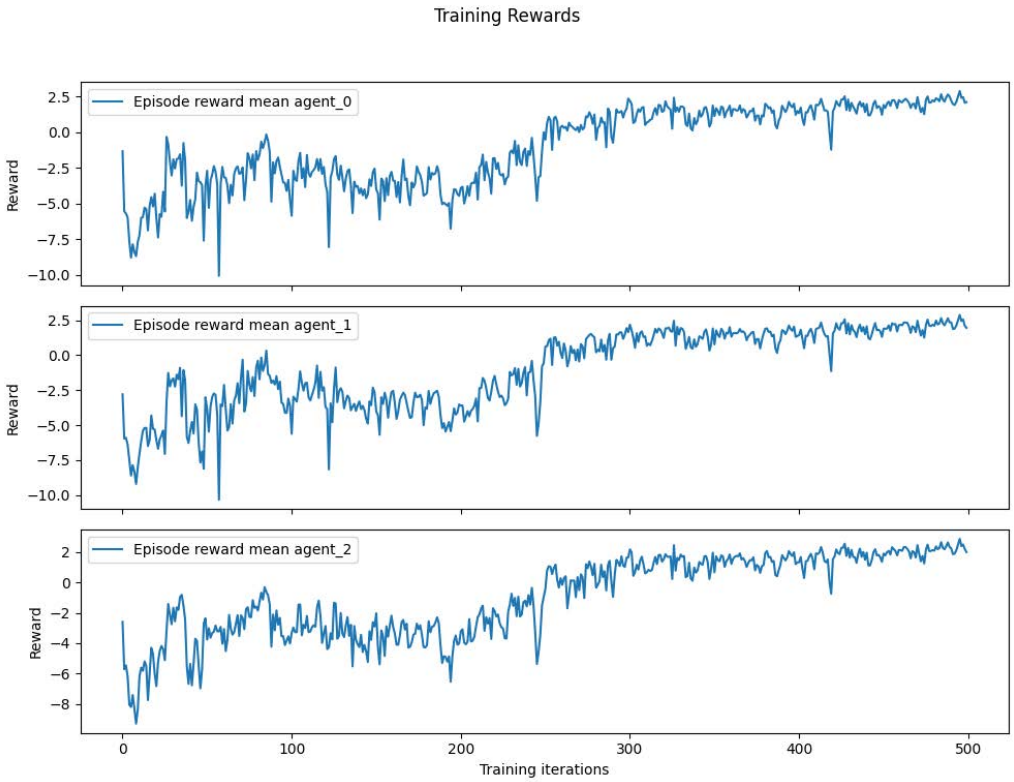

Figure 1: Agents performance after modifying a scalar hyper-parameter.

References

[1] Cover image designed by freepik

[2] Ryan Lowe, Yi I. Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. arXiv:1706.02275