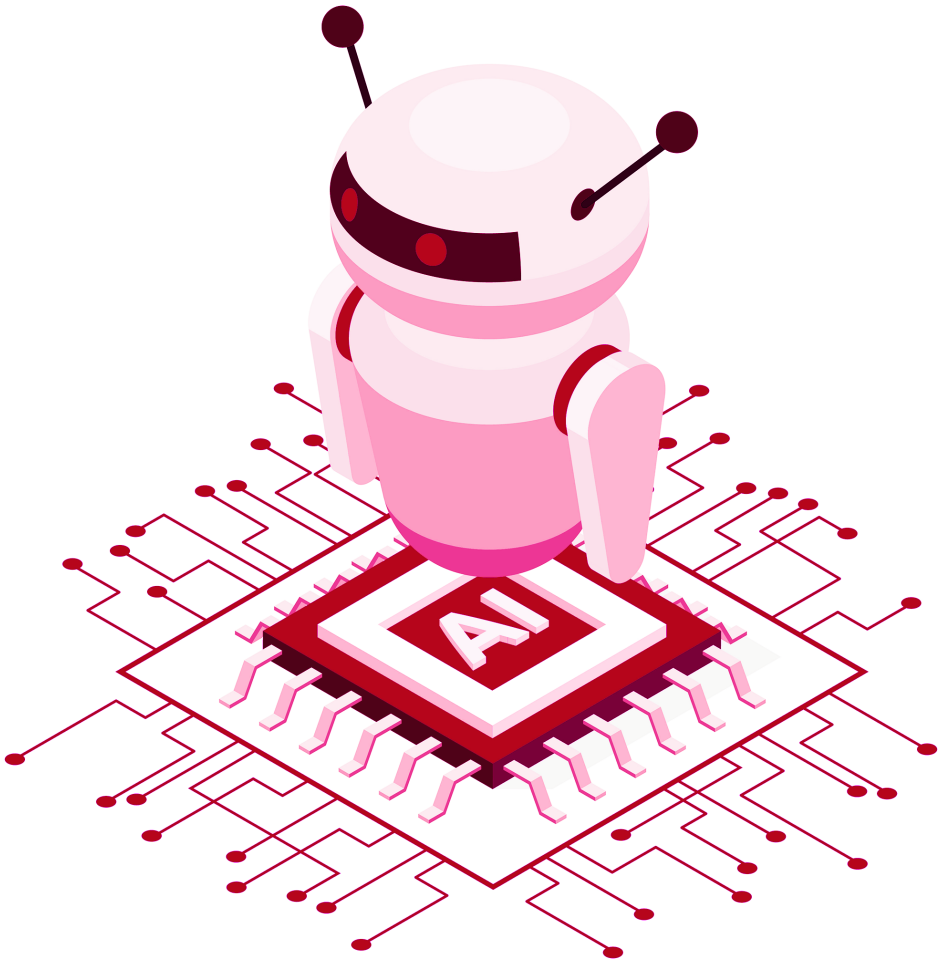

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Solution for Homework 13:

Multi-Agent RL

By:

Ayeen Poostforoushan

401105742

RIML

Spring 2025

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

Contents

1 Part 1: Game Theory Problems

The answers and theoretical explainations of this part is in the notebook.

2 Part 2: Implementing MADDPG/IDDPG

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.)

   In DDPG, the critic learns by estimating the value of the next state using the actor's actions. If we used the main policy network directly for this, the target would keep changing every time the actor updated. This would mean the critic is always chasing a moving target, which makes learning unstable and can lead to divergence.

   To fix this, DDPG uses target networks that update slowly with soft updates. Because these target networks change gradually, the critic learns toward a stable reference instead of something that shifts too quickly. This stability is important for training, as it helps both the actor and critic improve smoothly without destabilizing each other.

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.

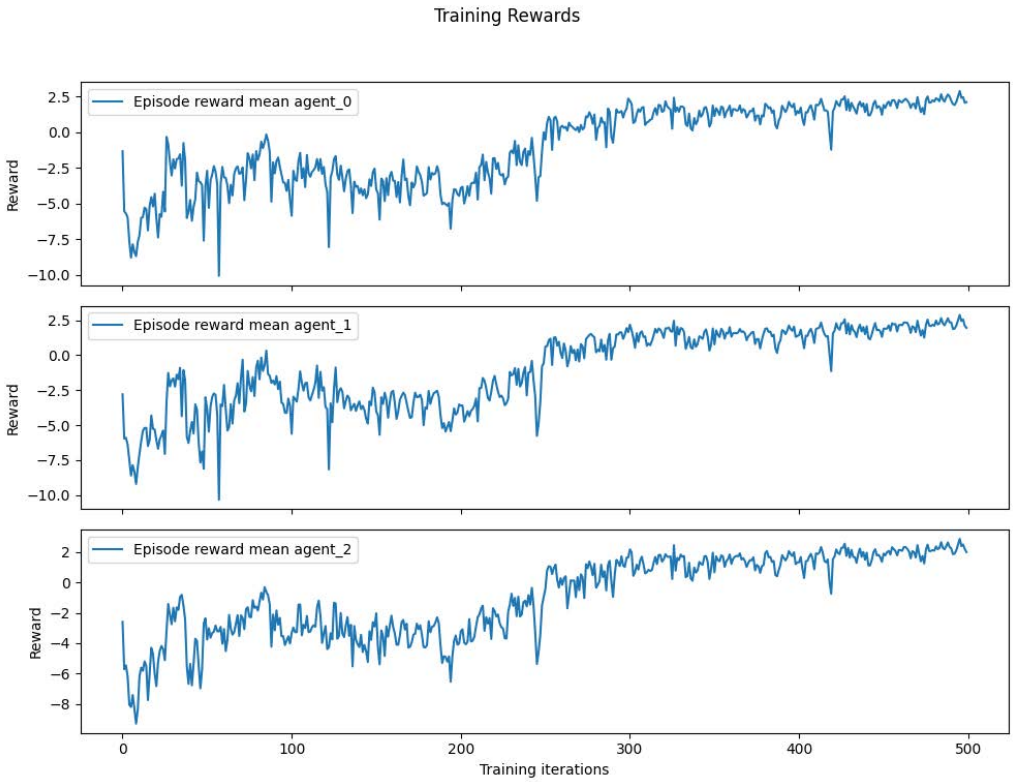

Figure 1: Agents performance after modifying a scalar hyper-parameter.

(a) The plots show the same pattern for all agents: at the start the rewards jump a lot and are mostly

low (negative), then after a couple hundred iterations the rewards rise and become much smoother and positive. In short — training is very noisy and poor at first, then the agents recover and learn.

(b) The changed scalar is most likely the exploration noise (e.g. sigma_init or noise amount). This noise makes actions random: high noise gives lots of exploration but makes early rewards noisy and low; reducing the noise over time lets agents exploit and get higher, stable rewards. Quick fixes: lower the initial noise or make the noise decay faster.

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