Distributed MARL Predator-Prey

Luca Fabri

Alma Mater Studiorum - University of Bologna

December 17, 2024

Table of Contents

Introduction

Distributed MARL Predator-Prey is a Python application that allows the user to train and simulate a Multi-Agent Predator-Prey environment through Reinforcement Learning.
After a training or simulation process, the user can also visualize the agents' movements inside an environment, or plot the error tendency of the latest training phase.

Table of Contents

Training process

The training of the environment is accomplished through:

- The implementation of a distributed technique inspired by Mava framework [Koc+23];
- The implementation of the MADDPG algorithm [Low+20].

MADDPG - Multi-Agent DDPG

MADDPG is the Multi-Agent version of DDPG. Each agent inside the environment consists of an Actor and Critic neural networks. By training these two networks, the agent can ultimately take actions, given an observation of the local space.

MADDPG - Multi-Agent DDPG

MADDPG implements a **centralized training, decentralized execution** framework:

- Critic. This neural network is responsible for evaluating how good the action decided by the Actor is, to guide it towards decisions that maximize future rewards. The Critic network is leveraged during training and it's centralized: it takes in input the joint states and actions of all agents;
- Actor. It learns the policy. It is decentralized: given the agent's observation, it decides what action should be taken.

Microservice architecture

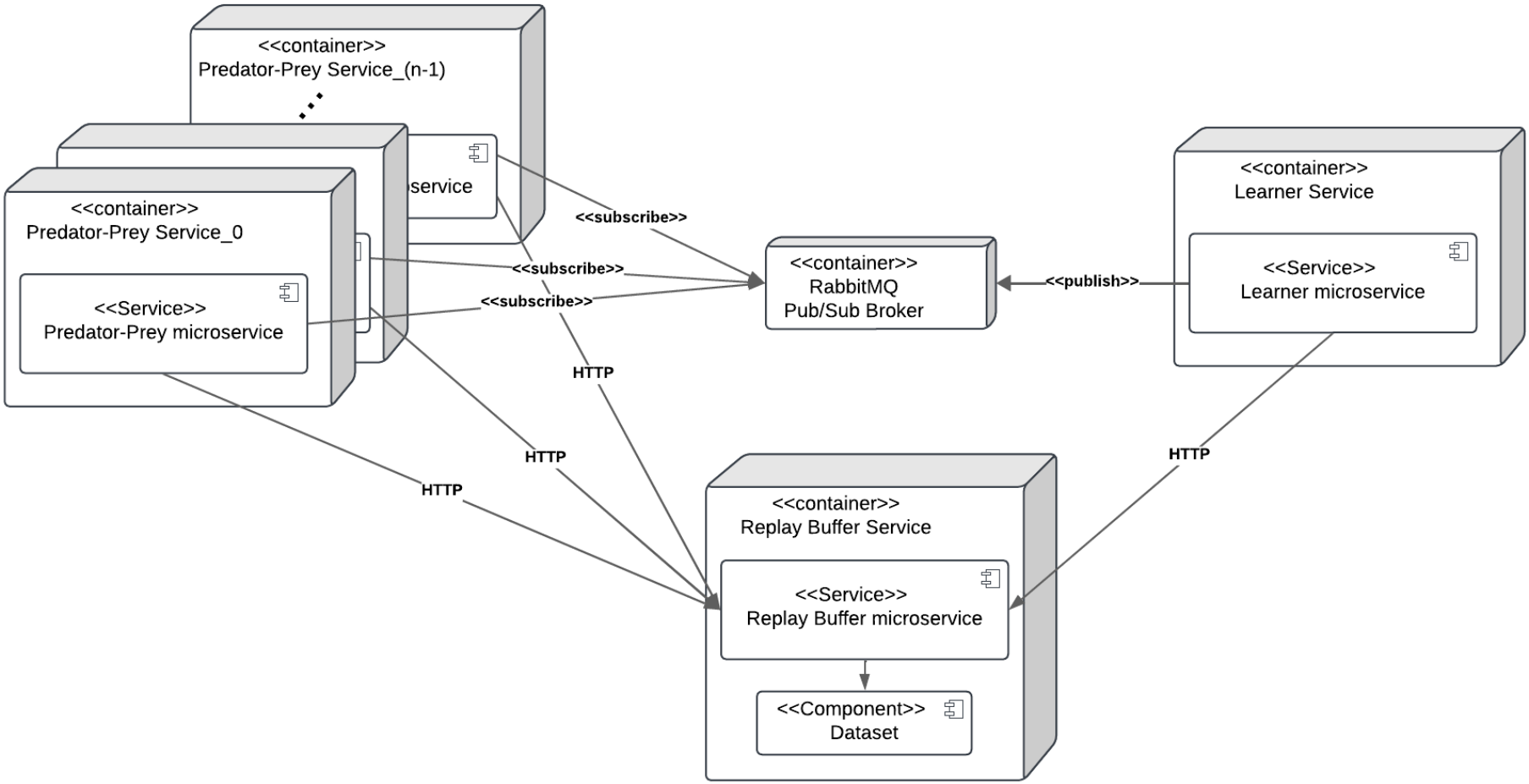

Figure: Deployment view of training phase

Microservice architecture

The System has been decomposed into three microservices:

- **Predator-Prey microservice**. It is responsible for managing the single environment. It stores agent experiences inside the distributed Replay buffer and subscribes for policy updates using a Pub/Sub channel;
- **Replay Buffer microservice**. Contains the dataset with the agent experiences belonging to the different distributed environments. It exposes a simple and intuitive ReST API to get and store data;
- **Learner microservice**. Implements the MADDPG algorithm. It batches data from the remote Replay buffer and publishes the Actor networks to each distributed environments by leveraging a Pub/Sub channel.

Table of Contents

Predator-Prey environment

Each distributed environment is a Predator-Prey mixed Cooperative-Competitive environment.
Both predators and preys have the same observation space, comprising a local view of the surrounding area. However, they have a different reward function:

- Prey's reward increases as the closest predator moves away;
- Predator's reward increases as the closest prey gets closer;

Predator-Prey environment

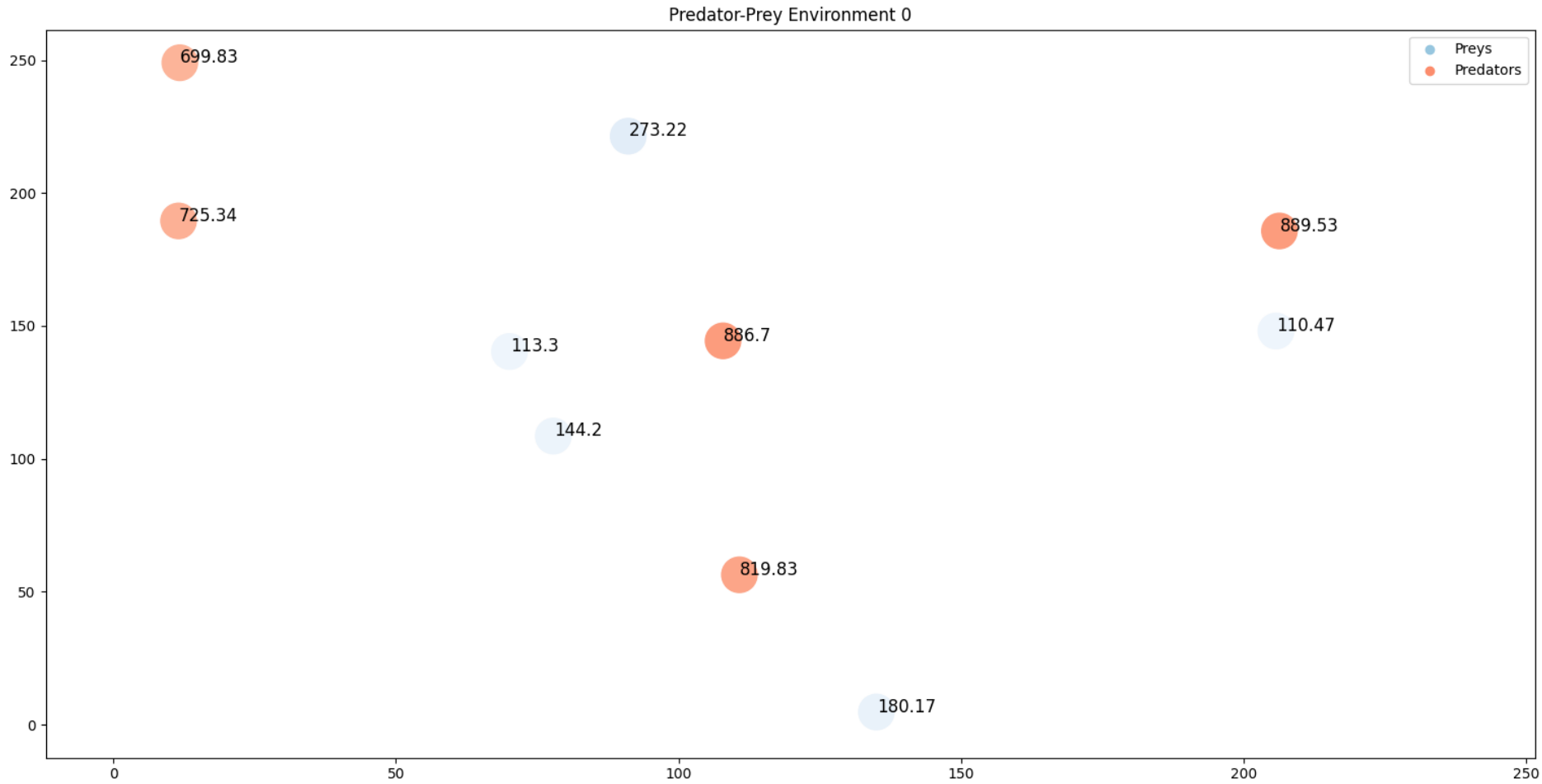

Figure: Animation's frame

Table of Contents

Experiments

The System has been trained for:

- 7h 12m 3s, in a Manjaro Linux 24.1 OS, using a CPU Intel(R) Core(TM) i7-8550U @ 1.80GHz and 16GB RAM;
- Using 5 predators and 5 preys, 5 environments in parallel;
- It doesn't converge. The non-convergence of may result from multiple factors: insufficient data / algorithm parameters.

Experiments

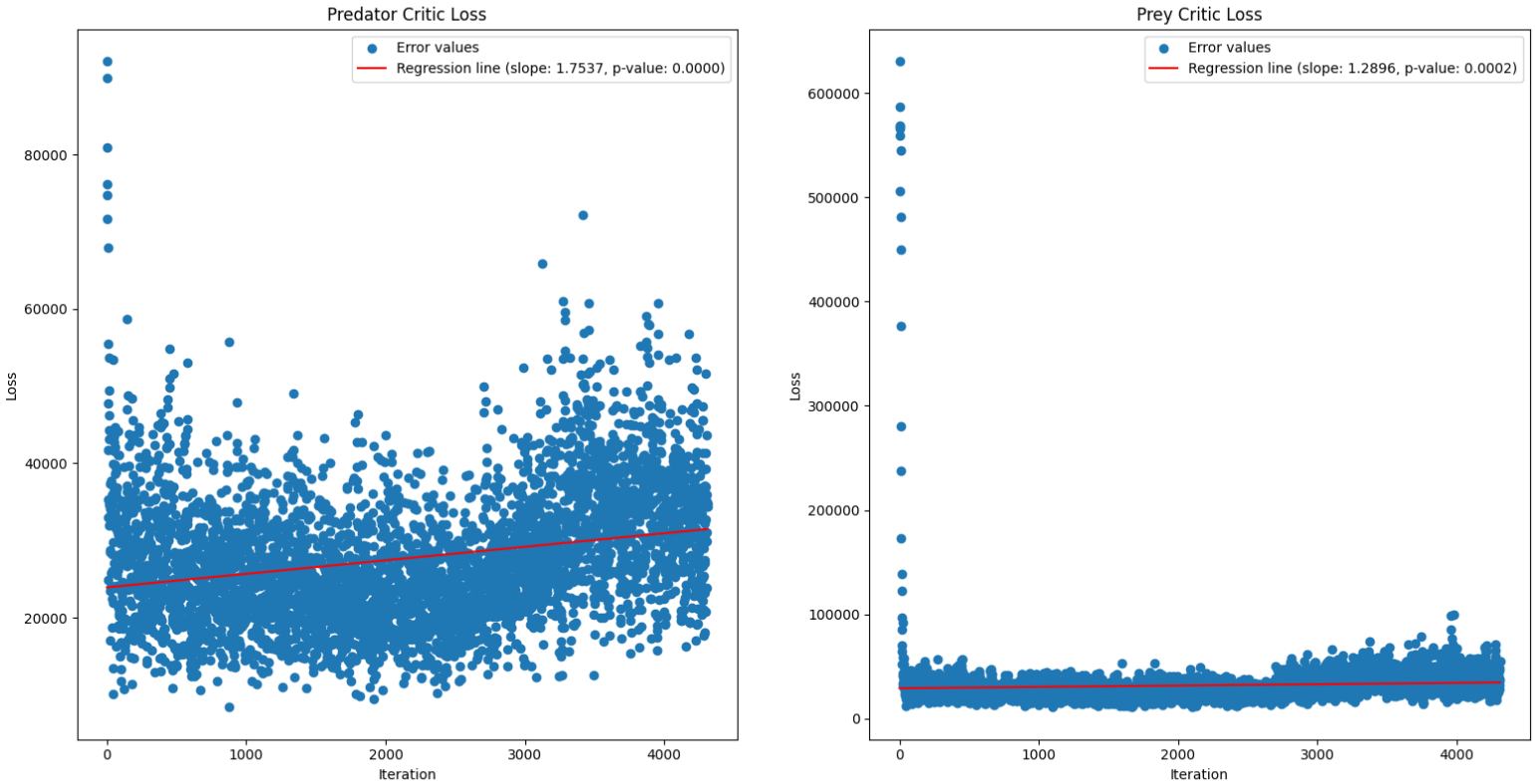

Figure: Critic network loss

Table of Contents

References I

[Koc+23] Ruan de Kock et al. *Mava: a research library for distributed multi-agent reinforcement learning in JAX*. 2023. arXiv: 2107.01460 [cs.LG]. URL: https://arxiv.org/abs/2107.01460.

[Low+20] Ryan Lowe et al. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. 2020. arXiv: 1706.02275 [cs.LG]. URL: https://arxiv.org/abs/1706.02275.