Northeastern University

Masters Project Course

Continuous Macro-Action Learning in Multi-Agent RL Environments

Author:
Atharva Wandile
(002110295)

Advisor:
Prof. Christopher Amato
Enrico Marchesini

8th February 2023

Contents

1 Abstract

Hierarchical methods provide a useful structured framework to solve asynchronous reinforcement learning problems efficiently, they also have the added benefit of transferring knowledge between similar tasks. However most of these approaches have been developed for single-agent environments with discrete action spaces, this limits their potential in real-world settings where most problems require the use of multiple agents working together asynchronously and the action spaces are usually continuous. Hence this project aims to explore the performance of the continuous macro-action algorithms on standard multi-agent reinforcement learning environments that will be extended to the macro-actions framework. The results of this research will provide valuable insights into the capabilities of macro-action methods in multi-agent, continuous action space environments and inform future developments in the field.

Keywords: Multi-Agent, Hierarchical ,Macro-Actions, Continuous

2 Introduction

Reinforcement learning has recently shown a lot of promise in spaces like robotics, simulations, games, etc. using actor-critic methods. However, most of these methods assume synchronous primitive actions and single agent settings. This does not scale well into the real world requiring asynchronous execution and temporally extended actions.

Due to the above issues hierarchical methods were developed for better transfer between tasks and a more abstract representation of state and action spaces. Besides this, they also naturally represent complex tasks in an abstracted and human understandable manner. For example waypoint navigation, and robotic tasks like grasping, placing etc. can be explained to a human rather than primitive actions like hundreds of fine motor controls, muscle contractions, sensors, etc.

The options framework is a well-known approach to adding temporal abstraction to an environment's actions. This is useful because it does not require prior knowledge of the environment and can learn both the action and the options from scratch without human input. However, it suffers from very high training variance and does not work well for multi-agent and partially observable environments. An alternative is to use a macro-action based frameworks which can be extended with Macro-Action Decentralized Partially Observable Markov Decision Process (MacDec-POMDP). The MacDec-POMDP is a general model for multi-agent problems with partial observability and different action durations. As a result, agents can start and end macro-actions at different time steps so decision-making can be asynchronous.

This project assumes that each domain has been predefined with a set of continuous macro-actions defined by a human, however, there has been some work on learning macro-actions from scratch that can be added to the future scope of the project. The continuous macro-actions will be tested on multiple custom domains for benchmarking; like macro-action versions of Multi Particle Environments MPE (Lowe et al., 2017) environments, and Box Pushing (Seuken and Zilberstein, 2012) domain for robotics. These will then be compared to the state-of-the-art primitive action methods. Initially, the goal is to design these environments for macro-action algorithms, then the next task is the implement the continuous macro-action algorithms on these environments and measure the performance.

3 Literature review

The use of macro-actions for learning asynchronously will require a different approach to maintain and update replay buffers. (Xiao et al., 2020) shows 2 approaches in centralized and decentralized policies. The decentralized one is called Macro-Action Concurrent Experience Reply Trajectories (Mac-CERTs). In this approach, each agent maintains its own macro-action and reward and updates are done only based on the individual agents' macro-action terminations. The centralized approach is called Macro-Action Joint Experience Replay Trajectories (Mac-JERTs). In this approach, a joint reward is collected at each time step and updates take place when any of the agents' macro-actions terminate.

These approaches were tested on (a) Capture Target, a variant of an existing multi-agent-single-target (MAST) domain (b) Box Pushing, a benchmark Dec-POMDP domain and (c) Warehouse Tool Delivery Domain inspired by human-robot interaction. The results show superior performance to primitive action alternatives and the centralized version outperforms the decentralized one.

Actor-critic methods and variants (Mnih et al., 2016; Schulman et al., 2017) are very well known and have shown state-of-the-art performance in most single agent tasks, however, they struggle to perform well in multi-agent asynchronous tasks. This paper (Xiao et al., 2022) shows an approach to apply actor-critic methods to multi-agent domains in three training paradigms : decentralized, centralized and centralized training for decentralized execution (CTDE). Formally they make use of the MacDec-POMDP framework for macro-action learning. A MacDec-POMDP is defined by a tuple $\langle I, S, A, M, \Omega, \zeta, T, R, O, Z, H, \gamma \rangle$ where I is a set of agents; S is the environmental state space; A is the joint primitive-action space over each agent's primitive-action set A_i; M is the joint macro-action space over each agent's macro-action space M_i; Ω is the joint primitive-observation space over each agent's primitive-observation set Ω_i; ζ is the joint macro-observation space over each agent's macro-observation space ζ_i; $T(s, a, s') = P(s'|s, a)$ is the environmental transition dynamics; and $R(s, a)$ is a global reward function. The objective of solv-

ing MacDec-POMDPs is to find a joint high-level policy Ψ that maximizes the value, $V^{\Psi}(s(0)) = E[\sum_{t=0}^{H_1} \gamma^t r(s(t), a(t))|s(0), \pi, \Psi]$, where $\gamma \in [0, 1]$ is a discount factor, and H is the number of (primitive) time steps until the problem terminates, π is the low level primitive policy for achieving a macro-action and s(0) is the start state. The three approaches shown by the paper are:

- Macro-Action-Based Independent Actor-Critic (Mac-IAC)
- Macro-Action-Based Centralized Actor-Critic (Mac-CAC)
- Macro-Action-Based Independent Actor with Centralized Critic (Mac-IACC)

  This last method (Mac-IACC) has 2 approaches:

  - Naive Mac-IACC
  - Independent Actor with Individual Centralized Critic (Mac-IAICC)

Mac-IAC takes a simple approach of assuming each actor behaves independently and learns its own macro-action policy and critic. Each agents accesses its own trajectories and performs updates using the on-policy TD gradient when the agents macro-action terminates. Hence the policy gradient is:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\Psi}_{\vec{\theta}}}\left[\nabla_{\theta_i} \log \Psi_{\theta_i}(m_i \mid h_i)\left(r_i^c + \gamma^{\tau_{m_i}} V_{\mathbf{w}_i}^{\Psi_{\theta_i}}(h_i') - V_{\mathbf{w}_i}^{\Psi_{\theta_i}}(h_i)\right)\right]$$

where r_i^c is the agent i's reward for macro-action m_i

Mac-CAC is a fully centralized training approach that behaves as a single joint agent to learn a centralized policy and critic for all agents. The policy gradient is :

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\Psi_{\theta}}\left[\nabla_{\theta} \log \Psi_{\theta}(\vec{m} \mid \vec{h})\left(\vec{r}^{c} + \gamma^{\vec{\tau}_{\vec{m}}} V_{\mathbf{w}}^{\Psi_{\theta}}(\vec{h}') - V_{\mathbf{w}}^{\Psi_{\theta}}(\vec{h})\right)\right]$$

where r^c is the cumulative reward for joint macro-action m

Naive-MAAC directly adapts the CTDE approach from AC to macro-actions by learning a joint macro-action value function and each agents policy gradients. Hence the critic has global information from all the agents but each agents policy update happens when its macro-action terminates.

However, the problem with the above approach is that the cumulative reward is based on when*any* agent finishes a macro-action. This is technically incorrect as the agent i's policy gradient does not estimate the value of executing macro-action m_i and will lead to higher variance in learning due to noise from other agents. the proposed solution to this was to use a separate centralized critic for each agent. The policy gradient then becomes:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\Psi}_{\vec{\theta}}}\left[\nabla_{\theta_i} \log \Psi_{\theta_i}(m_i \mid h_i)\left(r_i^c + \gamma^{\tau_{m_i}} V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}') - V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x})\right)\right]$$

Here the cumulative reward and value function is a more accurate prediction for the agents macro-action and leads to reduction in variance.

These approaches were tested on the Box Pushing and Overcooked domains. The results show that macro-action learning outperforms primitive actor-critic approaches. Further out of all the approaches Mac-CAC and Mac-IAICC seem to perform consistently better than the others in all the domains.

While these approaches use the original policy gradient update, we will be using this architecture and extending it to the Trust Region Policy Optimization(TRPO) (Schulman et al., 2015) algorithms (specifically PPO (Schulman et al., 2017)) and hope to get similar performance improvement for this project.

4 Initial Results

The first task was to create macro-action wrappers for the Multi Particle Environments (MPE) (Lowe et al., 2017) and test them on the primitive action algorithms to make sure we get the same results for consistency.

The cooperative environments were already created along with one adversarial environment(Simple Tag). Using this the goal was to complete the other adversarial environments. These environments are :

1. Simple Push
2. Simple Adversary

The main challenge was to understand the code base and all the implementation details and then apply the same to the other environments. One of the details causing initial issues was the use of absolute coordinates of the agents and adversaries for observations, as opposed to the relative coordinates used in the official MPE documentation (Lowe et al., 2017). Another was to understand the use of cost functions and their role in multi-goal-oriented and constrained RL. However, after getting over these initial roadblocks the environment wrappers have been created and are ready for testing. Some screenshots of the rendered environment can be seen in the figures.

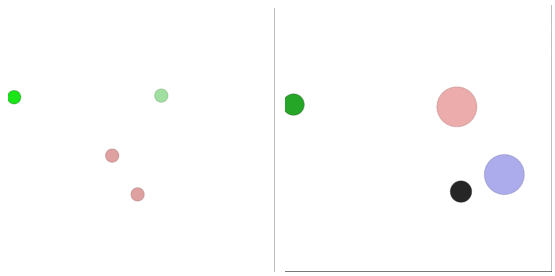

Figure 1: Simple Push and Simple Adversary Rendered Run

5 Project Plan

5.1 Short Term Goals

With access to the Northeastern Discovery cluster, the training time of most of these algorithms will be sped up by a significant time. Keeping that in mind these are the achievable goals within the next month.

1. Test MPE wrappers using primitive action algorithms
2. Test MPE wrappers using Continuous Macro-Action algorithms
3. Try to improve the results by hyperparameter tuning

5.2 Long Term Goals

These goals may or may not be completed by the end of the project however in an ideal world this would be the next step in the project.

1. Try different macro-actions to see the effect on performance
2. Extend to other multi-agent environments
3. Try to learn macro-actions while learning (optional) (Randlov, 1998)

References

Lowe, R., Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu (2016). Asynchronous methods for deep reinforcement learning.

Randlov, J. (1998). Learning macro-actions in reinforcement learning. In M. Kearns,

S. Solla, and D. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11. MIT Press.

Schulman, J., S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel (2015). Trust region policy optimization.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017). Proximal policy optimization algorithms.

Seuken, S. and S. Zilberstein (2012). Improved memory-bounded dynamic programming for decentralized pomdps. *CoRR abs/1206.5295*.

Xiao, Y., J. Hoffman, and C. Amato (2020, 30 Oct–01 Nov). Macro-action-based deep multi-agent reinforcement learning. In L. P. Kaelbling, D. Kragic, and K. Sugiura (Eds.), *Proceedings of the Conference on Robot Learning*, Volume 100 of *Proceedings of Machine Learning Research*, pp. 1146–1161. PMLR.

Xiao, Y., W. Tan, and C. Amato (2022). Asynchronous actor-critic for multi-agent reinforcement learning.