PantheonRL: A MARL Library for Dynamic Training Interactions

Bidipta Sarkar*, **Aditi Talati***, **Andy Shih***, **Dorsa Sadigh**

Department of Computer Science, Stanford University
{bidiptas, atalati, andyshih, dorsa}@stanford.edu

Abstract

We present PantheonRL, a multiagent reinforcement learning software package for dynamic training interactions such as round-robin, adaptive, and ad-hoc training. Our package is designed around flexible agent objects that can be easily configured to support different training interactions, and handles fully general multiagent environments with mixed rewards and n agents. Built on top of StableBaselines3, our package works directly with existing powerful deep RL algorithms. Finally, PantheonRL comes with an intuitive yet functional web user interface for configuring experiments and launching multiple asynchronous jobs. Our package can be found at https://github.com/Stanford-ILIAD/PantheonRL.

Introduction

Multiagent reinforcement learning (MARL) is becoming increasingly important as more AI systems are being deployed. Many potential applications of MARL involve dynamic interactions between agents, such as agents adapting to each other, ad-hoc coordination, and more (Fig 1). However, experimenting with these dynamic interactions using modern deep RL frameworks can be a difficult process. Existing MARL libraries are largely designed around training a fix set of agents, making them unsuitable for experimenting with more dynamic and adaptive agent interactions.

We propose PantheonRL, an easy-to-use and extensible MARL software package that focuses on dynamic interactions between agents. The goals of our package are:

1. to support adaptive MARL, with dynamic training interactions ranging from self-play, round-robin, adaptive (few-shot), and ad-hoc (zero-shot) training,
2. to build on top of existing powerful deep RL libraries, in particular StableBaselines3 (SB3) (Raffin et al. 2019),
3. to provide a web user interface for launching and monitoring experiments, with support for the different dynamic training interactions described above.

Related Work

There exists a number of MARL libraries, such as Mava (Pretorius et al. 2021), PyMARL (Samvelyan et al. 2019), and PettingZoo (Terry et al. 2020). The first two offer a selection of centralized multiagent training algorithms (QMIX, VDN, MADDPG), and PettingZoo hosts a large collection of multiagent environments. More related to our work is the combination of PettingZoo with RLLib (Liang et al. 2018), which provides some level of customization of agents. However, to the best of our knowledge, there does not exist a library designed around flexible agent objects for adaptive MARL. In contrast to previous work, our package prioritizes the modularity of agent objects. Each agent object is equipped with its own replay buffer and learning algorithm, allowing users to easily mix and match agents, swap their roles, and finetune individually for adaptive MARL.

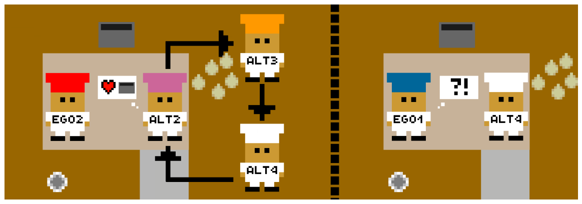

Figure 1: PantheonRL is designed to support dynamic training interactions, e.g., round-robin (left) or cross-play (right).

PantheonRL Framework

One of our desiderata is to build upon powerful single agent reinforcement learning (SARL) libraries. Our main design choice is thus: how do we interface with SARL algorithms, which train a policy network with input/output designated by the state/action space of an OpenAI Gym (Brockman et al. 2016) environment? Directly running a SARL algorithm on top of a joint multiagent environment (with joint observations and joint actions) is possible, but this produces a monolithic joint policy network that is undesirable for our first desiderata – to support adaptive MARL with dynamic training interactions.

Instead, our design is to split the training of each of the n agents as separate SARL instances, so that we produce n cleanly distinct policy networks that can be composed or finetuned for downstream adaptive MARL tasks. We next describe how we designed the environments and agents in a way that supports an intuitive API.

*These authors contributed equally.

Joint / Projected Environments Existing multiagent environments are generally defined as a *joint environment*, with a Gym `step` function handling n actions and observations. Each joint environment implicitly defines n *projected environments* that are Hidden-Parameter MDPs (Doshi-Velez and Konidaris 2016). The i-th projected environment handles the i-th agent's actions and observations, and has hidden-parameters characterized by the policies of the other $n-1$ agents.

Given a joint environment, we require only specification of the state/action spaces of the n projected environments. PantheonRL then automatically links with SARL algorithms to produce individual agent policies for each agent.

Ego / Partner Agents PantheonRL differentiates between an ego agent and the other $n-1$ partner agents. Each agent is equipped with its own replay buffer and learning algorithm. A critical design feature is that each agent's learning algorithm can be chosen from off-the-shelf SB3 algorithms, such as PPO, without any modifications.

We distinguish the ego agent role for two reasons. First, when doing round-robin training or adaptation, we often want to fix the ego agent and sample partners from a pool of possible partners. Second, to provide an intuitive API similar to that of SB3, we use the ego agent as the entry-point to the training procedure of all agents. In other words, ego.learn() will step through the environment, which triggers the learning algorithm of all the partner agents. The triggers are implemented so that all agents reuse the same joint trajectories, to avoid naively collecting the joint trajectories n times.

We highlight again the importance of Pantheon's compatibility with SB3 in giving our framework great flexibility. Each agent can specify its own learning algorithm imported directly from SB3. Designing our API around the ego agent also gives us access to many of the single-agent logging, debugging, and monitoring utilities of SB3.

Web User Interface

In addition to full-fledged command-line invocations, PantheonRL provides an easy-to-use web interface for launching and monitoring experiments. The web interface is valuable for prototyping MARL experiments, especially the dynamic training interactions supported by our package. This minimizes PantheonRL's initial user overhead, which is often non-trivial for other MARL packages. A demo of the user interface can be found at `https://youtu.be/3-Pf3zh_Hpo`.

The website guides the user in selecting the experiment parameters in stages – first configuring the environment, and then configuring each individual agent. The parameters are presented as dropdown menus, check boxes, buttons, and more, which help reduce a user's mental load during selection. The configurations also allow the user to save/load agent policies and trajectories for later experiments.

Not only is the website visually intuitive, it is also highly functional. One of the main features of the website is its asynchronous design. Built on top of Flask, the website allows a user to launch multiple asynchronous training jobs in the background. After the training jobs have launched, the website can either pull lightweight logging information to display to the user, or spawn a full Tensorboard service in the background to give the user complete monitoring capabilities. Moreover, the website stores a user's session information in a database, and supports login/logout if the user wants to manage multiple sessions.

Listing 1: Example round-robin python training script

```
env = gym.make('OvercookedMultiEnv-v0')
partner_1 = PPO.load(partner_1_file)
env.add_partner_agent(StaticPolicyAgent(
    partner_1.policy))
partner_2 = A2C('MlpPolicy', env)
env.add_partner_agent(OnPolicyAgent(
    partner_2))
ego_a = PPO('MlpPolicy', env, verbose=1)
ego_a.learn(total_timesteps=500000)
```

Listing 2: Example adaptation script

```
env = gym.make('OvercookedMultiEnv-v0')
env.add_partner_agent(StaticPolicyAgent(
    partner_2.policy))
ego_b = PPO.load(ego_b_file, env=env)
ego_b.learn(total_timesteps=500000)
```

Dynamic Training Interactions

Here, we demonstrate the simplicity of our API when training dynamic MARL interactions. The examples are on a 2-player environment, but they extend to n-player environments just as easily. First, we train an ego agent in a round-robin style by pitting it against two partner agents (Listing 1). Partner 1 is loaded from a previously trained PPO policy, but we wrap it as a `StaticPolicyAgent` so that its policy does not update anymore. Partner 2 will update its policy using A2C, and the ego agent will update its policy using PPO.

Next, we will evaluate partner adaptation by loading a previously trained ego agent, and pairing it with the partner 2 agent we just trained (Listing 2). Since we want the ego agent to adapt to the partner, we set partner 2 to not update its policy. Other training paradigms like ad-hoc pairing and standard MARL can be specified in a similar fashion.

PantheonRL provides a concise API while allowing for great flexibility in customizing training interactions, including the pool of partner agents, the toggling of partner updates, and the training algorithm for each individual agent.

Discussion

With focus on adaptive MARL and dynamic training interactions, PantheonRL is a valuable addition to the MARL software ecosystem. The modularity of the agent policies combined with the inheritance of StableBaselines3 capabilities together give users a flexible and powerful library for experimenting with complex multiagent interactions. To top it off, our intuitive yet functional web user interface, equipped with clean visuals and asynchronous job launches, makes PantheonRL a suitable library for a wide range of users.

Acknowledgments

We would like to acknowledge Office of Naval Research, and Air Force Office of Scientific Research for their support.

References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR*, abs/1606.01540.

Doshi-Velez, F.; and Konidaris, G. D. 2016. Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*.

Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

Pretorius, A.; ab Tessera, K.; Smit, A. P.; Eloff, K.; Formanek, C.; Grimbly, S. J.; Danisa, S.; Francis, L.; Shock, J.; Kamper, H.; Brink, W.; Engelbrecht, H.; Laterre, A.; and Beguir, K. 2021. Mava: A Research Framework for Distributed Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2107.01460*.

Raffin, A.; Hill, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; and Dormann, N. 2019. Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043.

Terry, J. K.; Black, B.; Grammel, N.; Jayakumar, M.; Hari, A.; Sulivan, R.; Santos, L.; Perez, R.; Horsch, C.; Dieffendahl, C.; Williams, N. L.; Lokesh, Y.; Sullivan, R.; and Ravi, P. 2020. PettingZoo: Gym for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2009.14471*.