Multi-Agent Learning with Prior Data

Aaron Jenson
aajenson@cs.washington.edu

Abstract

Multi-agent reinforcement learning (MARL) is an important area of research with applications in fields such as robotics, game theory, and traffic management. MADDPG is a popular algorithm for MARL, but it can suffer from issues related to efficiency and scalability, particularly when dealing with complex multi-agent environments. In this paper, I apply ideas used to improve single agent RL to MADDPG in order to improve MADDPG's efficiency and scalability through the use of offline data during online training. I evaluate the approach using a benchmark environment and show the results, demonstrating the potential of offline data for enhancing MARL.

1 Introduction

1.1 Background and Motivation

When humans can't achieve a task by themselves, usually one of the first approaches to completing the task is to call several people and work together to get it done. In robotics, we often take a different approach. Coordinating multiple robots to interact with each other proves to be a difficult task, so typically we build our robots bigger and more capable instead of relying on multiple smaller, cheaper robots to get the work done. This approach can work well in controlled environments such as factories, where despite the multiple robots on the factory floor, the whole system is essentially one large robot controlled by a single system.

However, this approach is difficult to scale and impractical for noisy environments. Consider a team of robots playing soccer [3]. Multiple robots can scale easily to various field sizes and player counts, which could not easily be done with a single robot. In addition, many industries that are difficult to automate, such as construction, rely heavily on cooperation between agents. For robotics to become an effective tool in these industries, we'll need to have tools to train robots to cooperate in the real world and not just in simulation.

Currently, MADDPG [7] is one of the most common algorithms for multi-agent reinforcement learning. It uses the approach of a centralized critic with decentralized actors for each agent, and works for cooperative or competitive environments. However, it hasn't seen significant use in the real world. Online reinforcement learning algorithms require a very large number of samples to train, making it difficult to effectively use outside of simulation. The added complexity of multi-agent RL makes it even more impractical.

1.2 Offline Data for Enhancing Online Training

For this reason, in this paper I take approaches that have been used to improve standard RL and apply them to MADDPG in hopes of making MADDPG more practical for real world use. In their recently released paper, Ball et al. [1] show that by combining offline and online data in the training process standard RL algorithms can be improved by up to 2.5x. Their work focuses on the Soft Actor-critic algorithm, but can be applied to other algorithms as well. This approach is agnostic to the quality of the offline data provided, making it trivial to gather this offline data.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In this paper I will show that by using offline data combined with REDQ [2], MADDPG can acheive significantly better results while also being much more sample efficient. I also provide results from several different combinations of ideas from Ball et al. [1], showing that Symmetric Sampling and REDQ help to improve training efficiency, while Layer Normalization impedes progress.

2 Related Work

There are several alternative multi-agent reinforcement learning approaches. One of the simplest is just training each agent individually with no knowledge of what the other agents are doing, the approach used by de F. Bassani et al. [3] in their paper on creating a team of robot soccer agents.

Other algorithms include MAA2C [6], MAPPO [10], and COMA [4], all of which are similar to MADDPG in that they are decentralized at test time and are based on standard RL algorithms. Another alternative allows communication between agents [5], which allows better performance but adds complexity to the system.

3 Methodology

Algorithm 1 MADDPG with Prior Data

Initialize offline data buffer, LayerNorm, Ensemble Size E, Gradient Steps G, and Critic targets to subset Z
Initialize Actor network and E Critic networks for each agent
for episode = 1 to M **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial state x
    for $t = 1$ to max-episode-length **do**
        if $t <$ num-random-episodes **then**
            for each agent i, select action $a_i = \mu_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
            Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathtt{x}'$
        else
            Execute random actions and observe reward r and new state $\mathtt{x}'$
        end if
        Store $(\mathtt{x}, a, r, \mathtt{x}')$ in replay buffer $\mathcal{D}$
        $\mathtt{x} \leftarrow \mathtt{x}'$
        for $g = 1, G$ **do**
            for agent $i = 1$ to N **do**
                Sample a random minibatch b_R of $\frac{S}{2}$ samples $(\mathtt{x}^j, a^j, r^j, \mathtt{x}'^j)$ from $\mathcal{D}$
                Sample a random minibatch b_D of $\frac{S}{2}$ samples $(\mathtt{x}^j, a^j, r^j, \mathtt{x}'^j)$ from offline data buffer
                Combine b_R and b_D to form batch b of size S
                Set $y^j = r_i^j + \gamma Q_i^{\mu'}(\mathtt{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \mu'_k(o_k^j)}$
                Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j (y^j - Q_i^{\mu}(\mathtt{x}^j, a_1^j, \ldots, a_N^j))^2$
                Sample and take the minimum of Z Critics θ_{iz}
                Update actor using the sample policy gradient:

$$\nabla_{\theta_{iz}} J \approx \frac{1}{S}\sum_j \nabla_{\theta_{iz}} \mu_i(o_i^j) \nabla_{a_i} Q_i^{\mu}(\mathtt{x}^j, a_i^j, \ldots, a_i, \ldots, a_N^j)|_{a_i = \mu_i(o_i^j)}$$

            end for
        end for
        Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

    end for
end for

Table 1: Simple World Comm V2 characteristics

Leader Adversary	Always knows location of agents Can send messages to adversaries Rewarded for contact with Agents
Adversaries	Rewarded for contact with Agents
Agents	Rewarded for distance from Adversaries Rewarded for distance to Food
Obstacles	Barriers for movement
Food	Provide reward for Agents
Forests	Hides location of Agents from Adversaries

3.1 MADDPG Overview

Multi-agent Deep Deterministic Policy Gradient [7] is a reinforcement learning algorithm designed for cooperative or competitive (or mixed) environments. Its main innovation is the concept of using a centralized training process that can take advantage of the observations of all agents, but at test time using a decentralized process that uses only the observations of that agent. To accomplish this, the critic in the actor-critic training method takes in observations from all agents, but the actor uses only the agent's observations. Lowe et al. [7] also proposes optionally using an ensemble of policies to make each agent more robust to changes in the strategy of other agents. Another optional part of the algorithm involves each agent learning an approximation of the other agent's policies, allowing them to use that information in their own decisions.

MADDPG has successfully been shown to work well in simulated environments, and works better than training each agent using DDPG without the centralized training process [7]. Though it works in simulation well, Suh et al. [9] argues that the reality gap is larger for MADDPG than single-agent RL because the system is much more complex and MADDPG fails to place constraints that ensure that its execution is actually decentralized.

3.2 Offline Data and Online Training with MADDPG

Ball et al. [1] recently released a paper describing a method of using offline data to augment online training. This algorithm, called RLPD (Reinforcement Learning with Prior Data), is agnostic to the quality of the data and is shown to significantly improve training times over purely online training methods. Though RLPD was developed using SAC (Soft Actor-critic) as its base algorithm, its concepts are applicable to other algorithms, and specifically are compatible with DDPG and by extension, MADDPG.

RLPD has three core design choices that allow it to use offline data efficiently, along with a few optional elements that help performance in some environments. I integrate the core ideas into MADDPG in this paper, and leave the optional elements as further work. First, I add symmetric sampling which takes half of the data for each batch from online data, and half from offline data. Next, I use layer normalization to keep Q-values at reasonable levels. Lastly, I use random ensemble distillation to make each update more efficient. Each of these strategies comes directly from Ball et al. [1].

The full algorithm used is described in Algorithm 1. [1]

3.3 Benchmark Environments

To evaluate the effectiveness of this approach, I use the `simple_world_comm_v2` benchmark environment. The environment agents and features are described in Table 1. Offline data was collected by training first using the standard MADDPG [7] algorithm, then evaluating the trained policies for several episodes and collecting the actions and rewards. To fully understand and evaluate the effectiveness of this approach, more training environments need to be used.

[1] Implementation details can be found at https://github.com/aaronjenson/maddpg

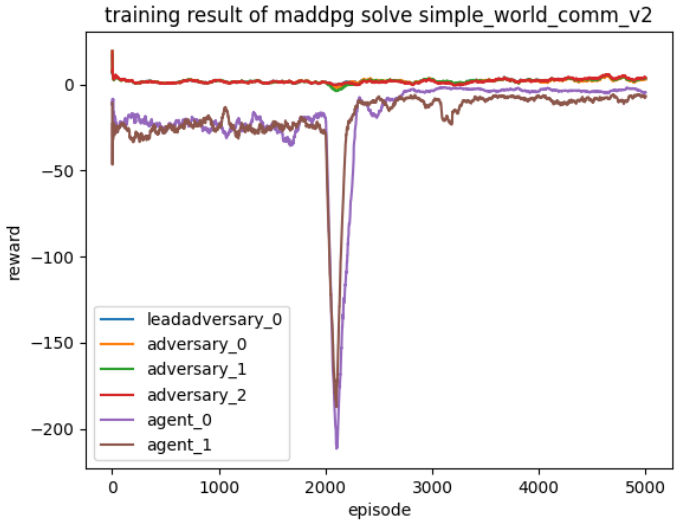

Figure 1: Standard MADDPG results.

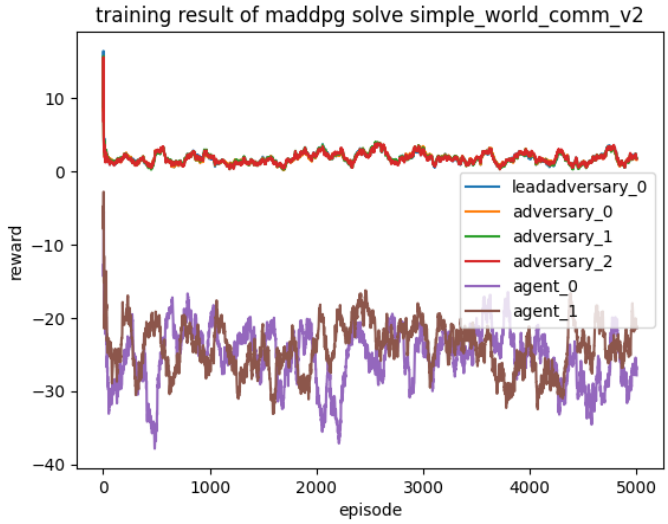

Figure 2: MADDPG with offline data, layer normalization, and REDQ.

4 Results and Analysis

Figure 1 shows the results of running basic MADDPG in the test environment. As shown, the agent's score improves a slightly throughout the training process but plateaus quickly, and the adversary's score doesn't show any significant improvement. Note that the dip at 2000 episodes in the agent's reward coincides with the transition from completely random actions to actions chosen by the policy.

Figure 2 shows the results of adding in all of the discussed strategies. As shown, its only improvement over standard MADDPG is the lack of the dip after policy-chosen actions begins. In this sense, all three of these ideas together allow for the policy to quickly match the performance of completely random actions, but after that nothing seems to improve.

We contrast both these results with the results shown in Figure 3, using Symmetric Sampling and REDQ without Layer Normalization. In this example, we see that the adversaries start to improve

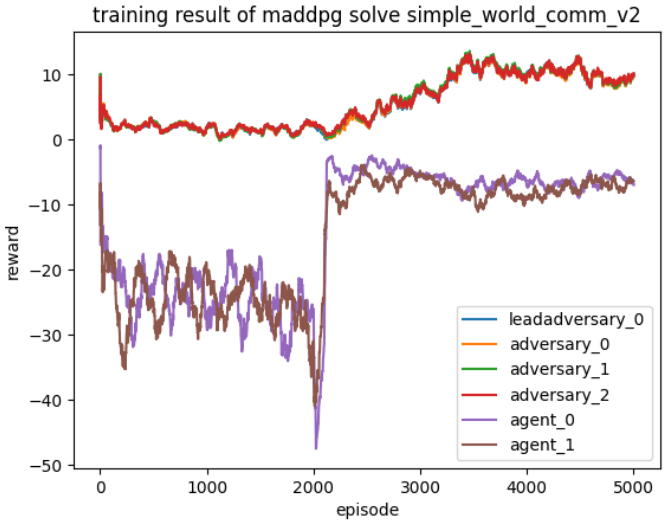

Figure 3: MADDPG with offline data and REDQ.

and the agents get a significant bump in performance. However, we still don't see much learning going on in the agents after the initial training started.

The poor performance when Layer Normalization is involved is of particular note in this experiment, as Lowe et al. [7] find it to be of high importance in their work. This may be due to my error, or may indicate an inherent difference in either the DDPG algorithm or in MARL that causes Layer Normalization to function worse than in the paper cited. In any case, this indicates that perhaps by finding a different method of constraining Q-values, performance could be improved further.

5 Conclusion and Future Work

5.1 Summary of Proposed Approach and Effectiveness

In this paper, I showed that by taking advantage of offline data in online training, we can improve MADDPG's performance. The proposed improvements include sampling half the data from each batch from online data and the other half from offline, using layer normalization to keep Q-values low, and using random ensemble distillation to make each update more efficient. I show that while layer normalization is not an effective strategy in this case, symmetric sampling and REDQ are both helpful in reducing the number of samples needed to train multiple agents.

5.2 Limitations and Future Directions for Research

While this approach improved standard MADDPG, there is still work to be done, especially on training in the real world. Smith et al. [8] have shown that training robots from scratch in the real world is possible in a remarkably short time using several tricks to optimize the training process. Applying this to multi-agent setting would contribute greatly to the field.

Safety is also a major concern in multi-agent robotics. It's much harder with two or more robots to avoid collisions, so finding ways to minimize accidents during the training process would help allow us to focus on training in the real world.

Another avenue for improvement is making MADDPG more robust to the other agents, not just in their actions, but in the number of cooperative and/or competitive agents in the environment. In a standard MADDPG implementation, each agent will have its own policy, even if its goal is the same as other agents in the environment. This makes it hard to scale to larger environments with more agents, since training with different agents results in different policies. Finding a way to add or remove agents from the environment without needing to re-train the agents would also be potential

future work. In this work, I trained a separate actor and critic ensemble for each agent, even when some agents had the same observations, reward function, and actions available to them. Further work might explore using the same networks for each identical agent, allowing more samples to be collected for a single policy in less time.

5.3 Conclusion and Contributions

From this work, I conclude that there are still many opportunities for improving MARL out there. Through approaches such as the ones proposed in this paper, multi-agent learning can be made efficient enough to train in the real world, especially when combined with common sim2real practices. By improving the cooperative abilities of robots, we will be able to scale up the tasks that our current robots can accomplish without needing to redesign a robot from the ground up for each task we come across. While much more research is needed for this approach as well as other possible approaches, it is within our grasp to use multi-agent policies for practical applications on real robots in the near future.

Acknowledgments and Disclosure of Funding

Special thanks to Ahbishek Gupta for his instruction in CSE 599G, Deep Robotic Learning, at the University of Washington, which provided the author with their first real introduction to reinforcement learning. Thank you as well to the fellow students in the class who engaged in discussions and helped improve my understanding of these topics.

References

[1] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data, 2023. URL `https://arxiv.org/abs/2302.02948`.

[2] X. Chen, C. Wang, Z. Zhou, and K. W. Ross. Randomized ensembled double q-learning: Learning fast without a model. *ArXiv*, abs/2101.05982, 2021.

[3] H. de F. Bassani, R. A. Delgado, J. N. de O. Lima Junior, H. R. Medeiros, P. H. M. Braga, M. G. Machado, L. H. C. Santos, and A. Tapp. A framework for studying reinforcement learning and sim-to-real in robot soccer. *ArXiv*, abs/2008.12624, 2020.

[4] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *ArXiv*, abs/1705.08926, 2017.

[5] S. Han, S. Zhou, J. Wang, L. Pepin, C. Ding, J. Fu, and F. Miao. A multi-agent reinforcement learning approach for safe and efficient behavior planning of connected autonomous vehicles. 2020.

[6] S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments, 2017. URL `https://arxiv.org/abs/1706.02275`.

[8] L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *ArXiv*, abs/2208.07860, 2022.

[9] Y.-H. Suh, S. Woo, H. Kim, and D.-H. Park. A sim2real framework enabling decentralized agents to execute maddpg tasks. *Proceedings of the Workshop on Distributed Infrastructures for Deep Learning*, 2019.

[10] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *ArXiv*, abs/2103.01955, 2021.

A Appendix

A.1 Full Experimental Results

Below are all graphs from various experiments conducted, with labels of their unique characteristics.

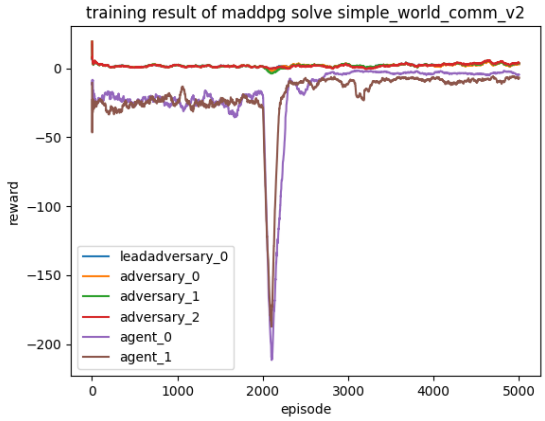

Figure 4: Standard MADDPG.

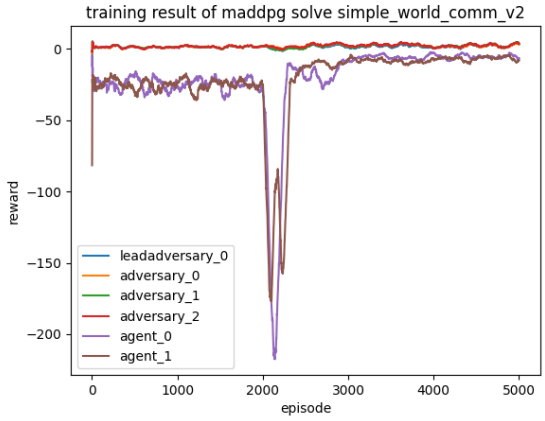

Figure 5: MADDPG with Symmetric Sampling.

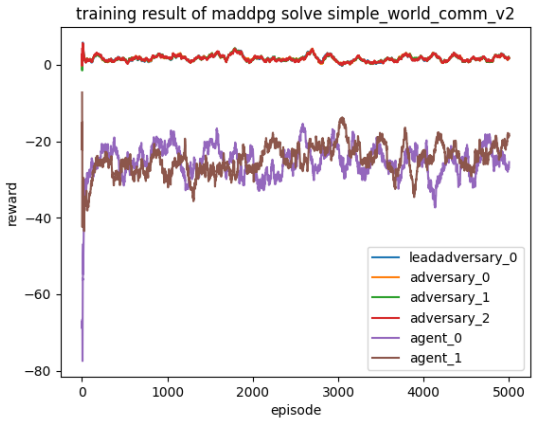

Figure 6: MADDPG with Symmetric Sampling and LayerNorm.

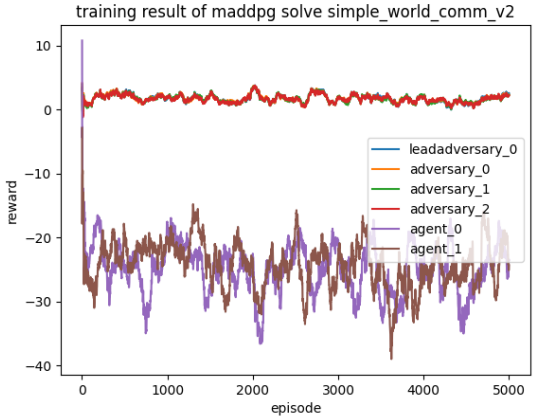

Figure 7: MADDPG with LayerNorm.

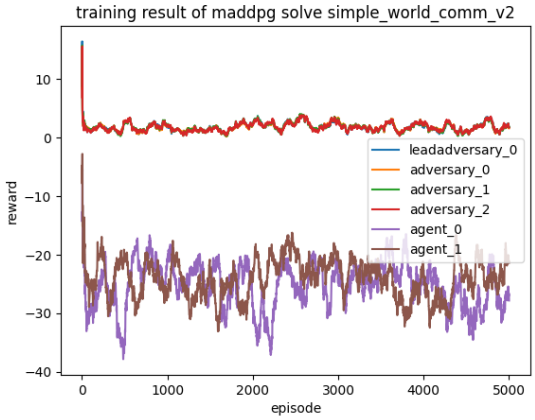

Figure 8: MADDPG with Symmetric Sampling, LayerNorm, and REDQ.

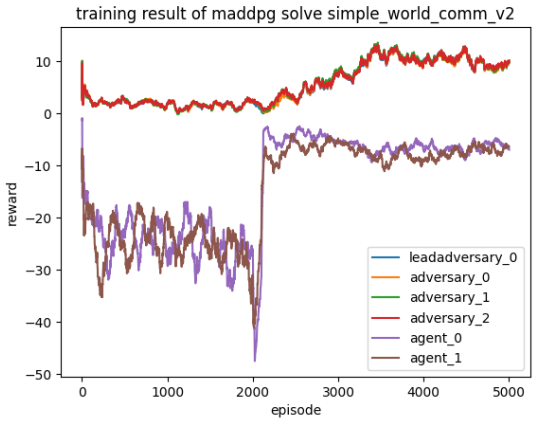

Figure 9: MADDPG with Symmetric Sampling and REDQ.

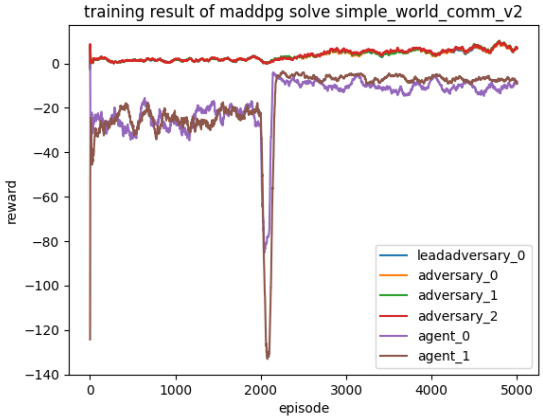

Figure 10: MADDPG with REDQ.

Figure 11: MADDPG with Symmetric Sampling and REDQ, longer training.

Figure 12: MADDPG with Symmetric Sampling and REDQ, longer training with higher quality offline data (gathered from the results of Figure 3).