Multi-Agent Reinforcement Learning based on Value Distribution

Ruiqun Li1**, Ruobing Wang1, Tao Tian2, Fukai Jia2 and Zhong Zheng2

1China Aerospace Academy of Systems Science and Engineering, Beijing, 100048, China
2Aerospace Hongka Intelligent Technology (Beijing) CO., LTD., Beijing, 100048, China
1551140018@xzyz.edu.cn **Corresponding Ruiqun Li’s e-mail: ruiqunli@163.com

Abstract. Reinforcement learning in a multi-agent setting is very important for real-world applications, but it brings more challenges than those in a single-agent environment. In the multi-agent setting, the agent generally has a bias of overestimation on the value function. In our work, we pay attention to the issue of overestimation bias with continuous actions in the multi-agent learning environment. We propose a method to reduce this bias by adopting the distributional perspective on reinforcement learning. We combine it within the framework of off-policy learning Actor-Critic and propose a novel approach Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG). We empirically evaluate it in three competitive and cooperative multi-agent settings. Our results show that in a series of difficult motor tasks the agents trained by MAD3PG significantly outperforms existing benchmark.

1. Introduction

In recent years, multi-agent systems are solving many real-world environment issues. Due to the recent improvement of single agent deep reinforcement learning (DRL), which led to the success of robotics control[1], playing go[2] and playing Atari game[3]. The application of DRL algorithms to multi-agent learning settings has been a rising trend. A lot of works have achieved great success on various issues, including multiplayer games[4], traffic control[6] and smart-grid applications[7].

It generally suffers from the training instability when applying single agent deep reinforcement learning algorithms to multi-agent environment. Because of the agent's policy is changing, the environment is non-stationary from the perspective of any single agent. When using the single agent deep reinforcement learning algorithm, this may cause the scale of the action space to increase exponentially relative to the number of agents. To solve this instability issue, one method is to suppose independent learners[8], where the agent sees the influence of other agents as part of the training environment. Because of other agents’ behavior changes with time, the transition probability will also change, which violates Markov hypothesis.

Therefore, one approach is to break up these systems into independent and dispersed agents in the training process, and update them in the centralized execution phase, thereby maintaining the Markov properties in the execution process. So, this paradigm avoids challenges of non-stationary environments and non-Markov in the learning process. In spite of using a centralized critic can stabilize training, such as the counterfactual multi-agent (COMA) policy gradients[9], and the multi-agent deep deterministic policy gradient (MADDPG)[10], the learned policies can still be fragile to its training partners.
In this work, we focus on addressing bias of overestimation and put forward a new algorithm, Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG). MAD3PG extends the classic MADDPG algorithm[10]. Its core idea is that utilizing a distributional[11] version of the critic function, which offers a more stable, better learning signals. We tested our introduced MAD3PG algorithm in three mixed competitive and cooperative settings, and experimental results showed that the agent trained by MAD3PG algorithm overcomes baseline results in the three settings.

2. Related Work
Multi-agent reinforcement learning (MARL)[12] is a challenging problem in the field of AI. As deep learning-based reinforcement learning methods become more and more popular, they have been used to the MARL environment[13]. The recent work in DRL is to utilize deep learning to approximate agent policy and action-value function. However, some applications of DRL approaches to MARL suffer some problems, such as non-stationary from the perspective of any independent agent[14] and lack of coordination in cooperative settings[15].

The most relevant to this work is to propose a method of centralized agent policy algorithm ground on actor-critic policy gradient [16]. Lowe et al. [10] extended the DDPG algorithm to a multi-agent environment and proposed the MADDPG algorithm. Shariq Iqbal and Fei Sha[17] proposed an actor-critic algorithm, which uses a centralized computing critic with a shared attention mechanism to train decentralized policies in a multi-agent environment. S. Li et al. [18] proposed M3DDPG, which approximates a minimax training target by using hostile interference to the action of other agents when updating critics and policies. Ackermann J et al. [19] investigated the problem of value function bias of overestimation on the multi-agent environment and proposed MATD3 algorithm which uses double centralized critics. Our MAD3PG algorithm is built on top of MADDPG and the core idea of MAD3PG is that utilizing a distributional version updated by critics to provide more stable and better learning signals.

3. Background

3.1. Markov Games
In this work, we think about an extension of Markov decision processes (MDPs), called Markov games[20], to multi-agent settings. A Markov game for N agents is defined by a set of states $S$, a set of actions $A_1, ..., A_N$ and a transition function $T: S \times A_1 \times ... \times A_N \rightarrow D(S)$. Each agent obtains reward as a function of the state and agent's action $r_i, S \times A_1 \times ... \times A_N \rightarrow R$, which relies on action of all agents. Each agent $i$ uses a random policy to choose action $\pi_i: O_i \rightarrow Dist(A_i)$, where $O_i$ is a different observation set for decentralized agents. Every agent has the goal of maximizing their total expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i$, where $0 < \gamma < 1$ is a discount factor.

3.2. Q-Learning and DQN
Q-Learning and DQN[21] are very popular approaches in deep reinforcement learning and have been used in multi-agent environments [22]. Q-learning is an off-policy algorithm that makes use of an action-value function for policy $\pi$ as $Q^\pi(s, a) = E[R \mid s_t = s, a_t = a]$. The DQN that uses multi-layer perceptrons (MLPs) to approximate the Q-function learns the value function corresponding to the best policy by minimizing the loss:

$$L(\theta) = E_{s,a,r,s'}[(Q^* s, a \mid \theta) - y]^2$$

(1)
where
\[ y = r + \gamma \max_{a'} Q(s', a') \]

\( Q_{\theta'} \) is a target Q function with parameters \( \theta' \) slowly follow the network parameters of \( Q_{\theta} \), which helps stabilize learning. Additionally, tuples \((s, a, r, s_0)\) are saved in an experience replay buffer \( D \).

### 3.3. Deterministic Policy Gradient (DPG) Algorithms

In 2014, Silver et al. [23] proposed deterministic policy gradient (DPG). They also showed that in some areas, deterministic policies learn more faster than random policies.

In [24], a Deep Deterministic Policy Gradient (DDPG) algorithm was proposed, which is an reinforcement learning algorithm based on DPG continuous control problems. It uses transitions from an experience replay buffer \( D \) to perform off-policy updates and also utilizes a target network. Using this, we can write the gradient as:

\[ \nabla_{\theta} J(\theta) = E_{s \sim D} [\nabla_{\theta} \mu_{\theta}(s)] \nabla_{a} Q_{\theta}(s, a) \bigg|_{a = \mu_{\theta}(s)} \] (2)

where \( D \) is the experience replay buffer.

### 3.4. Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

The core idea of the MADDPG algorithm [10] is using decentralized execution and centralized training method to learn centralized critics with access to all agent policies. More concretely, the centralized Q-function is updated with

\[ y = r + \gamma Q_{\mu}(x', a_1', ..., a_N') = \frac{(Q_{\mu}(x, a_1, ..., a_N) - y)^2}{\nabla_{\theta} J(\theta)} \] (3)

The deterministic policy of agent \( i \) can be updated by adopting the Q-function:

\[ \nabla_{\mu} J(\mu) = E_{x, a, r, x} [Q_{\mu}(x, a_1, ..., a_N) \bigg|_{a_i = \mu_i(a_i)}] \] (4)

where \( \mu_i \) is the policy of agent, \( a_i \in X \) represents the observation value of agent when taking action and \( x \) is the full state information. In decentralized execution process, each agent policy \( \mu_i \) uses only local information \( a_i \) to generate an action.

### 4. Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG)

In this part, we propose our novel algorithm, Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG), which is based the MADDPG algorithm and aims to reduce the bias of overestimation.

#### 4.1. Categorical distribution

After Bellemare et al. [25], we think about the categorical parameterization, using a discrete distribution parametrized by \( N \) and \( V_{\min}, V_{\max} \in \mathbb{R} \). The distribution is defined on a fixed set of atoms \( z_i \) and

\[ \Delta z = \frac{V_{\max} - V_{\min}}{N - 1} \]

there was hyperparameters with the number of atoms \( N \). Knowing these, the corresponds to the distance between atoms, and \( z_i = V_{\min} + i \Delta z : 0 \leq i < N \) gives the position of each atom. In a sense, these atoms are the "canonical returns" of our distribution. Then we may define the value distribution as:
\[ Z_\theta(x, a_1, ..., a_N) = Z_i, P_i(x, a_1, ..., a_N) := \frac{e^{\theta(x, a_1, ..., a_N)}}{\sum_j e^{\theta_j(x, a_1, ..., a_N)}} \]  

(5)

However, under the Bellman operator defined earlier, this distribution is not closed because scaling and adding values will no longer depend on the support of atomic definitions. This support is defined by the \((V_{MIN}, V_{MAX})\) hyperparameters. Therefore, we utilize the projection version that assigns the Bellman operator[25]. Letting \(m\) be the probability of the Bellman operator \(\Phi T\) of the projection distribution, which is applied to some target distribution \(Z_{\text{target}}\), we can describe the loss in the form of the cross entropy:

\[ d(\Phi T_\pi Z_{\text{target}}, Z) = \sum_i m_i \log P_i(x, a_1, ..., a_N) \]  

(6).

4.2. MAD3PG Optimization

The method used in our work starts with the MADDPG algorithm and makes some improvements. First, we think about the distributional critic proposed by Bellemare et al [25]. To introduce the distributional update we first review \(Q(x, a)\) in the form of a random variable \(Z_\pi\), such that \(Q(x, a) = E[Z_\pi(x, a)]\). Then the distributional Bellman operator is defined as:

\[ (T_\pi Z)(x, a) = r(x, a) + \gamma E[Z(x', \pi(x')) | x, a] \]  

(7)

One important component of stabilizing MADDPG algorithm is the use of target network, a target Actor network with parameters \(\theta'\) and a target Critic network with parameters \(\omega'\), using off-policy temporal difference to update the Critic network. Similarly, in the mad3pg algorithm, we utilize centralized training and decentralized execution settings. We assume that during the training period, we can access the past observations, actions, rewards and policies of all agent. To utilize the function in the actor-critic architecture described above, we can parameterize the distribution and define a loss. We may write the Critic network loss as:

\[ L(\theta) = E_{x,a,r,x'}[d(Y, Z_\pi(x, a_1, ..., a_N))], \]

\[ Y = r + \gamma Z'_\pi(x', a_1', ..., a'_N) | a'_1 = \mu'(o_i) \]  

(8)

where \(d\) is a measure of the distance between two distributions.

We may solve this distributional policy gradient algorithm by using the action-value distribution in the actor update. We can describe the gradient of the agent's expected return under the policy \(\mu_t\) as:

\[ \nabla_{\theta} J(\mu_t) = E_{x,a,r,x'}[\nabla_{\theta} \mu_t(a_t | o_t) \nabla_{\theta} Q_\pi'(x, a_1, ..., a_N) | a'_1 = \mu'(o_i)] \]

\[ = E_{x,a,r,x'}[\nabla_{\theta} \mu_t(a_t | o_t) E[\nabla_{\theta} Z_\pi'(x, a_1, ..., a_N) | a'_1 = \mu'(o_i)] \]  

(9)

**Algorithm** Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG)

for episode=1 to M do

  Initialize a random process \(\epsilon\) for action exploration

  Receive initial state \(x\)

  For t=1 to max-episode-length do

    For each agent i, select action \(a_t = \mu_\theta(o_t) + \epsilon\)

    Execute actions \(a = (a_1, ..., a_N)\) and observe reward \(r\) and new state \(x'\)

    Store \((x, a, r, x')\) in replay buffer \(D\)
\[
x \leftarrow x'
\]
For agent \(i = 1\) to \(N\) do

Sample a random minibatch of \(S\) samples \((x^k, a^k, r^k, x^k')\) from \(D\)

\[
Y_k = r^k + \gamma Z_i' (x^k, a^1, \ldots, a^N) |_{a^j = \mu^j(\phi^j)}
\]

Set

\[
\delta_a = \frac{1}{M} \sum_i \nabla a \delta(Y_k, Z_a(x^k, a^1, \ldots, a^N))
\]

Update critic by minimizing the loss

\[
\delta_a = \frac{1}{M} \sum_i \nabla a \delta(Y_k, Z_a(x^k, a^1, \ldots, a^N)) |_{a^i = \mu^i(\phi^i)}
\]

end for

Update actor using the sampled policy gradient:

\[
\theta_i' \leftarrow \tau \theta_i + (1 - \tau) \theta_i'
\]

\[
\omega_i' \leftarrow \tau \omega_i + (1 - \tau) \omega_i'
\]

end for

end for

5. Experiments

5.1. Environments

We test the efficacy of our method on the particle-world settings used to evaluate MADDPG in[10]. The particle world environment consists of two-dimensional world with continuous space, in which agent can exert a force on himself. We pay attention to the three competitive and cooperative environments to test the efficacy of our approach. They are shown in Figure 1.

![Figure 1](image_url)

**Physical deception** \(N = 2\) agents cooperate to arrive at a target landmark from \(L = 2\) landmarks. They will be rewarded based on how close one of them is to the target landmark, but if the opponent is very close to the target landmark, they will obtain a negative reward. However, the opponent (\(M = 1\)) also wants to reach the target landmark; but it does not know which landmark is the target landmark. Therefore, the cooperative agents learned to cover and extend all landmarks to deceive opponents.

**Cooperative navigation** \(N = 3\) agents will be rewarded based on their distance from each landmark. If agent conflict with other agents, they will be punished. Therefore, the agent must learn to cover all \(M = 3\) landmarks while avoiding collisions.
Predator-prey In this variation of the classic predator-prey game, $N = 3$ slower cooperative agents want to chase faster opponents ($M = 1$) in a randomly generated environment, while $L = 2$ large landmarks block up the road.

5.2. Comparison to MADDPG
We implement our method, named MAD3PG, and test the effect of the agent trained by our method and the agent trained by the MADDPG algorithm in each environment. Algorithm selects hyperparameters via grid search over mini-batch size $b = [256,1024]$, learn rate $\alpha = [0.01,0.003,0.001]$, policy update frequency $f = [1,2,3]$ and the number of atoms $N = [51,101]$. The hyperparameters of our experiment are $b = 1024$, $\alpha = 0.01$, $f = 2$, $N=51$. In the experiment, we utilize MLPs and LSTM to approximate the Q-functions and policies. At the same time we use $V_{\text{MAX}} = -V_{\text{MIN}} = 10$ to conduct our experiment.

The experimental results in three environments are demonstrated in Figure 2. The x-axis is the training episodes and y-axis is the reward of agents. The experimental results show that agents trained by MAD3PG can get higher reward that MADDPG approach on all the scenarios. Higher reward implies a better policy.

![Figure 2](image1.png)

Figure 2. Performances of MAD3PG (mad3pg, red) and MADDPG (maddpg, green), on physical deception, cooperative navigation, predator-prey from left to right.

![Figure 3](image2.png)

Figure 3. Performances of different Critic network (MLPs and LSTM), in physical deception, cooperative navigation, predator-prey from left to right on MAD3PG.

The experiment evaluates and analyzes the effects of different Critic network on the performance of the algorithm model and show the results in Figure 3. We can see, the MLPs showed the higher reward in physical deception and predator-prey, the LSTM showed the higher reward in cooperative navigation, resulting a higher final performance.

6. Conclusion
In our work, we introduce a new algorithm, Multi-Agent Deep Distributional Deterministic Policy Gradient (MAD3PG), for multi-agent reinforcement learning (MARL). It is ground on decentralized execution, centralized training method, reducing the bias of overestimation by adopting the distributional perspective on reinforcement learning. We have shown that MAD3PG outperforms the benchmark methods on three scenarios. It will be an interesting direction to explore robust and effective methods to solve multi-agent reinforcement learning problems. We take this as our future work.
Acknowledgement
This research was financially supported by project of Aerospace Hongka Intelligent Technology (Beijing) CO., LTD.

References
[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, et al. Learning dexterous in-hand manipulation. 2020, 39(1):3-20.
[2] Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587):484-489.
[3] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, Rusu Andrei A, Veness Joel, Bellemare Marc G, Graves Alex, Riedmiller Martin, Fidjeland Andreas K, Ostrovski Georg, Petersen Stig, Beattie Charles, Sadik Amir, Antonoglou Ioannis, King Helen, Kumaran Dharshan, Wierstra Daan, Legg Shane, Hassabis Demis. Human-level control through deep reinforcement learning. [J]. Nature, 2015,518(7540).
[4] Sun P, Sun X, Han L, et al. TStarBots: Defeating the Cheating Level BuiltIn AI in StarCraft II in the Full Game.[J]. arXiv: Artificial Intelligence, 2018.
[5] Berner C, Brockman G, Chan B, et al. Dota 2 with Large Scale Deep Reinforcement Learning[J]. 2019.
[6] Wu, C.; Kreidieh, A.; Parvate, K.; Vinitsky,E.; and Bayen, A. M. 2017. Flow: Architecture and bench-marking for reinforcement learning in traffic control. arXivpreprint arXiv:1710.05465.
[7] F. D. Li, M. Wu, Y. He, and X. Chen. Optimal control in microgrid using multi-agent reinforcement learning. ISA Trans., 51(6):743–751, 2012.
[8] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In ICML, pages 330–337, 1993.
[9] Foerster J, Farquhar G, Afouras T, et al. Counterfactual Multi-Agent Policy Gradients[J]. 2017.
[10] Ryan Lowe∗. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments[J]. 2017.
[11] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In International Conference on Machine Learning, pp. 449–458, 2017.
[12] Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews 38(2):156.
[13] Tampuu, A., Matisen, T., Kodelja, D., Kuzovkin, I., Korjus,K., Aru, J., Aru, J., and Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. PLoS One, 12(4): e0172395, April 2017.
[14] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In AAAI Conference on Artificial Intelligence, 2018.
[15] Mordatch, I. and Abbeel, P. Emergence of grounded com-positional language in multi-agent populations. In AAAI Conference on Artificial Intelligence, 2018.
[16] Peng, P.; Y uan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017a. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. CoRR abs/1703.10069.
[17] Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning[C]/International Conference on Machine Learning. 2019: 2961-2970.
[18] Li S, Wu Y, Cui X, et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient[C]/Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 4213–4220.
[19] Ackermann J, Gabler V, Osa T, et al. Reducing Overestimation Bias in Multi-Agent Domains Using Double Centralized Critics[J]. 2019.
[20] Littman M L. Markov games as a framework for multi-agent reinforcement learning[M]/Machine learning proceedings 1994. Morgan Kaufmann, 1994: 157-163.
[21] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, 2015, 518(7540): 529-533.
[22] Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016a. Learning to communi-cate with deep multi-agent reinforcement learning. CoRRabs/1605.06676.
[23] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]. 2014.
[24] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. Computer ence, 2015, 8(6): A187.
[25] Bellemare M G, Dabney W, Munos R. A distributional perspective on reinforcement learning[J]. arXiv preprint arXiv:1707.06887, 2017.