An Improved Agent Strategy Training Method Based on DIAL

Runzhi Liu, Baolong Guo, Qianying Li and Biyan Wang
School of Aerospace Science and Technology, Xidian University, Xi’an 710071, Shaanxi – P.R. China.
Email: rzl2019@foxmail.com; blguo@xidian.edu.cn

Abstract. The traditional multi-agent method has a slow convergence speed and needs to train many parameter groups. If no algorithm optimization is performed, it takes a lot of time to reach a large number in order to achieve optimal results in the whole process. We optimize the parameter settings in the traditional algorithm training process, accelerate the convergence speed of the intelligent body in learning, and achieve the convergence effect faster.

1. Introduction
In the era of science and technology, from the battlefield confrontation to the tactical strategy of all kinds of competitions, it is no longer entirely a human resource decision, but a new decision-making style. The rise of artificial intelligence, the establishment of new training methods and models have provided strong support for the strategic layer of robotic confrontation systems. In the defense system, the force deployment strategy of the combat system, the arrangement of battlefield reconnaissance and monitoring points, and the optimization of material resources transportation will all involve strategic decision-making research. The related technology has undergone rapid changes, from the popular AlphaGo to AlphaZero, using artificial intelligence methods to train decision-making methods. How to make robots get more efficient and timely strategic control in the competition, realize the intelligent robot confrontation system that truly learns and evolves rapidly, and it is also the potential for intelligent development in the future. This paper studies the agent strategy system in order to embody the empirical and evolutionary aspects of human thinking decision in the multi-agent strategy system.

The existence of the traditional robotic intelligence field: the incompleteness of the environment observation leads to the instability of the system, and the problem of learning the network is complicated and takes a lot of time. Most of the methods and systems are currently designed for the single agent control field and cannot effectively solve the above problems. Based on the integration of traditional classical algorithms, the robot self-confrontation system is designed to achieve the effect of the system. The reinforcement learning, convolutional neural network and recurrent neural network are used to train the robot’s strategic framework.

2. Related Work
2.1. Deep Q-Networks
In the case where the single agent system environment is known, the agent can obtain its current environmental state \( t \), unit discrete time, randomly select the strategy \( \pi \) and the current action \( u_t \in U \), and the next time state \( S_{t+1} \), the obtained reward signal is \( r_t \).

For maximum reward, \( R_t = r_t + \lambda r_{t+1} + \lambda^2 r_{t+2} + \lambda^3 r_{t+3} + \cdots \) (Where, \( t \) represents the reward obtained by the agent at time \( t \), and \( \lambda \) represents the discount factor \([0,1]\)). The Q-function of strategy \( \pi \) is
Q'(s, u) = E[R | s_i = s, u_i = u] and get the optimal action function Q'(s, u) = max_s Q'(s, u), then action function obey the be Bellman optimality equation Q'(s, u) = E_t[r + \lambda \max_{s'} Q'(s', u') | s, u]. The DQN[1] uses neural network parameters \theta to represent Q(s, t | \theta), which is minimized at the next iteration

L_i(\theta) = E_{s_i, a_i, r_i}[(y^{true} - Q(s_i, u_i | \theta))^2].

The behavior of the agent i is selected by the behavior selector from Q(s, u | \theta). The action selector typically uses the greedy strategy to obtain the maximum Q value, which selects the random value. DQN uses a pool of experience, during which the agent builds an empirical data set and then extracts a portion for training.

2.2. Deep Recurrent Q-Networks
In the deep Q learning network, it is assumed that the agent environment is fully observable, that is, the agent receiving environment S as an input. In contrast, in the case where the agent environment portion is observable, the environment information S is hidden, and the agent only receives the environment observation value O associated with S, and generally does not solve the environmental local observation problem.

Kunknecht and Stone[2] proposed a Deep Recurrent Q-Network to solve the local observation problem of a single agent. They propose that instead of a feedforward neural network to approach Q(s, u), a recurrent neural network is used to approximate Q(s, u). The deep loop Q network can maintain the internal state of the agent and count the results of the agent observation over time. The model can also be obtained by adding an additional input h_{t-1} (h_{t-1} indicates the hidden state of the network at time Q(o_t, h_{t-1}, u)).

2.3. Policy Gradient (PG) Algorithms
The main idea is to directly adjust the strategy parameters \theta of the agent and take the maximum value of the gradient \nabla_{\theta}J(\theta) in the direction of J(\theta). Its strategy gradient algorithm can be written as[3]:

\nabla_{\theta}J(\theta) = E[\nabla_{a} \log \pi_{\theta}(a | s)Q'(s, a)] (1)

The Agent Strategy Gradient method is used in several common algorithms, which differ in that the estimation method for the Q-function Q' is different. For example, a simple sample set return result R' = \sum_{t=0}^{T} \lambda^{t} r_t can be used in the enhancement algorithm[4], or an approximation of the real action function Q'(s, a) as in the time series learning. In the agent strategy gradient algorithm, Q'(s, a) is called a critic and is used in each Actor-Critic algorithm[5].

The energy strategy gradient method exhibits a high variance in the gradient calculation, and the problem is more apparent in the multi-agent field. Since the rewards finally obtained by the agent often depend on the common behavior of multiple agents, there is a lot of variability based on the behavior of a single agent itself (when the agent's optimization process does not need to consider the actions of other agents), thereby increasing the gradient of the agent. Next, optimize the algorithm in a multi-agent, so that the agent is exponentially decreasing in the appropriate gradient direction.

2.4. Deterministic Policy Gradient (DPG) Algorithm
The Policy Gradient algorithm (PG) can be extended to the Deterministic Policy gradient (DPG). Under certain conditions, we can write the gradient of the agent i as:

\nabla_{a}J(\theta) = E_{s' \sim \rho}(\nabla_{a} \log \pi_{\theta}(a | s)Q'(s, a) | s, a, u) (2)
Since the DPG algorithm relies on $\nabla Q'(s, a)$, it requires the action state set $a \in A$ of the agent to be continuous.

2.5. Deep Deterministic Policy Gradient Algorithms (DDPG)

DDPG\textsuperscript{[6]} is an extension of the deterministic policy gradient algorithm that uses a deep neural network to compute the agent strategy $\pi$ and the evaluation function $Q'$. DDPG is an off-policy algorithm. The strategy of the agent learning is different from the strategy of actual execution. It samples the motion track of the agent from the experience pool stored in the entire agent training. DQN\textsuperscript{[1]} is also used in DDPG as its target network.

2.6. Differentiable Inter-Agent Learning (DIAL)

DIAL\textsuperscript{[7]}, the method combines the intelligent body learning method with the Q network method. It is not only possible to share parameters between agents, but also to transfer gradients from one agent to another through a communication channel.

The method works by replacing the communication action with a direct connection between the output of one network to another during the learning of the agent. Therefore, when the communication message is discrete data during learning, the agent can freely send a real value message to other agents. These communication messages can in turn act as activation functions for other networks, so that gradients can be transmitted along the communication channel, enabling end-to-end backpropagation across agents.

As shown in Figure 1, Q-Net represents Q-Network, AS represents Action selector, O represents the observation of environment, m represents the information exchange between agents. C(green arrow)gradients flow across agents from the recipient to the sender.

![Figure 1. DIAL Information transmission process](image_url)

3. Methods

3.1. Messaging Protocol

In the original algorithm, at each moment, a greedy algorithm strategy is used to select an action for the agent, and a Q function\textsuperscript{[8]} is used to send a message to other agents.
The action $a_{t-1}$ of the agent step $t-1$, the index $\alpha$ of the agent $a$, and the observed environment $o_{t-1}^\alpha$, step $t-1$ the hidden state $h_{t-1}^{\alpha}$ inside the agent and the message $m_{t-1}^\alpha$ transmitted by the other agent are all transmitted to the agent at time $t$. After the agent selects the action, the current state of the agent and the bonus information obtained by the agent action are counted. When the agent reaches the last time or reaches the end state, for each agent $a$ with a time step of $j$, the agent's Q value $Q_j^a$ is calculated using the reward $r_j$ and the discount factor obtained by the agent. Finally, the cumulative gradient $\nabla \theta$ is calculated by Q-value regression.

$$Q(\alpha, m_{t-1}, h_{t-1}^\alpha, u_{t-1}^\alpha, a; \theta)$$

For the action performed by the agent $u_{t-1}^\alpha$, the Q value is $y_{t-1}^\alpha$, and the gradient chain $\lambda_{t-1}^\alpha$ of the agent communication message is also updated, and the gradient contains the deviation value of the error message $m_t^\alpha$ of the agent outgoing communication message $\sum (\Delta Q_{t-1}^\alpha)^2$.

3.2. Improved Messaging Protocol

$$Q(\alpha, m_{t-1}, h_{t-1}^\alpha, u_{t-1}^\alpha, a; \theta)$$

As in equation (5), it uses the step of $t-1$, the action $u_{t-1}^\alpha$, the index $\alpha$, the observed environment $o_{t-1}^\alpha$ and the hidden state $h_{t-1}^\alpha$ of agent. However, after the experiment, it is found that the convergence speed is still not fast enough. It takes a lot of time to train the model. So, we add a parameter to the formula to speed up the convergence of the agent. After experimentation, we find that the method is feasible. In the information exchange of agents, we add the relative direction $\hat{d}$ of the calculated agents.

The status of the delivery information after the addition is:

$$Q(\alpha, m_{t-1}^{\hat{\alpha}}, h_{t-1}^{\alpha}, u_{t-1}^{\alpha}, \hat{d}; a; \theta)$$

Figure 2. IDIAL information transmission process, in addition to the initial transmitted message, we also transmit the relative positional $\hat{d}$ relationship to the next agent.

The improved information flow chart of this article is shown as Figure 3, the red block diagram in the figure indicates the information transmission after joining add the relative direction $\hat{d}$ of the calculated agents. Then, step $t - 1$ send the message to the others agent at step $t$. 
4. Experiments Results

We perform experiments as Figure 4 on the original algorithm (DIAL), initial algorithm (RL), and improve algorithm (IDIAL) in the same environment, and statistically train 5000 loss function curves, as shown in the Figure 5.

From the loss function curve, we can see that the improved algorithm IDIAL proposed in this paper achieves convergence at around 1300 times, the DIAL algorithm converges around 1500 times, the RL algorithm converges around 3000 times, and the IDIAL algorithm can be more than the DIAL algorithm. Fast convergence and speed increased by about 13.3%.
We perform experiments on the original algorithm (DIAL), initial algorithm (RL)\cite{6}, and improve algorithm (IDIAL) in the same environment, and statistically train 5000 rewards value curves, as shown in the figure 6.

![Rewards function curve](image)

**Figure 6.** Rewards function curve

It can be seen from the agent reward value curve that when the training is 5000 times, the algorithm IDIAL proposed in this paper can obtain a higher reward value and quickly reach convergence.

5. **Conclusions and Future**

This paper gives a brief introduction of DQN, DRQN, PG Algorithms, DPG Algorithms and DDPG Algorithms. Then, the parameters passed in the DIAL model are analyzed in detail, and an improvement based on the application environment is proposed. The improved algorithm can converge faster than the original algorithm, and the reward value is higher. In the original algorithm, 1500 convergence is needed in the same experimental environment. The improved algorithm only needs 1300 times and the speed is increased by 13.3%.

6. **References**

[1] V Mnih, K Kavukcuoglu, D Silver, A A Rusu, J Veness, M G Bellemare, A Graves, M Riedmiller, A K Fidjeland, G Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[2] M Hausknecht and P Stone. Deep recurrent Q-learning for partially observable MDPs. arXiv preprint arXiv:1507.06527, 2015.

[3] R S Sutton, D A McAllester, S P Singh, and Y Mansour. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems, 1057–1063, 2000.

[4] R J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

[5] R S Sutton and A G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[6] T P Lillicrap, J J Hunt, A Pritzel, et al, continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, 2016.

[7] R Lowe, Y Wu, A T JeanHarb, Pieter, et al. Mordatch.Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv preprint arXiv:1706.02275, 2018.

[8] G Tesauro. Extending q-learning to general adaptive multi-agent systems. In Advances in neural information processing systems, 871–878, 2004.