Multi-agent Confrontation System Based on Reinforcement Learning

Qianying Li[a], Baolong Guo[b], Zhe Huang[c] and Cheng Li[d]

School of Aerospace Science and Technology, Xidian University, Xi'an 710071, China

[a]liqianying2020@foxmail.com, [b]blguo@xidian.edu.cn, [c]huangz@stu.xidian.edu.cn, [d]licheng812@stu.xidian.edu.cn

Keywords: Multi-agent, Adversarial System, Deep Reinforcement Learning, Actor-Critic Framework

Abstract: The development of machine learning technology has so far no longer relied entirely on artificial power in adversarial decision-making, but there has been an updated decision competition. The climax of the development of artificial intelligence and machine computing capabilities has provided powerful technical support and hardware support for the development of intelligent adversarial systems. The improved actor-critic extended multi-agent strategy gradient method (ACEM) is mainly used to optimize the behavior decision of multi-agent cooperation and hostility. This algorithm introduces extended strategy review information on the basic actor-critic framework for reward calculation. The use of adversarial strategy inference reduces the impact of environmental instability on strategy selection. This method has significantly improved the degree of completion of target tasks in cooperative and hostile environments.

1. Introduction

The development of machine learning technology has so far no longer depended entirely on artificial power in adversarial decision-making, but there has been newer decision competition. The climax of the development of artificial intelligence and machine computing capabilities has provided powerful technical and hardware support for the development of intelligent adversarial systems. In recent years, from the popular AlphaGo to AlphaZero, it has revealed new changes of the agent decision system. In order to enable the agent to achieve more efficient and real-time policy control in the confrontation, the agent confrontation has always been a hot field in the development of artificial intelligence. Currently the mainstream Agent policy systems mainly include Hierarchical Decision-making System (HDS) and Decision Tree-based Decision-making (DT-DM). HDS system is generally used in single agent training by combine scene segmentation and area calculation to select strategies from the overall height analysis. The DT-DM system uses multi-level layering to divide agent strategies, it is mainly used for policy selection when agents have weak correlations and a single environment. This article aims at the centralized strategy method of the agent and expands on the single agent strategy learning based on the deep reinforcement learning method. This paper completes multi-agent centralized strategy learning and independently executes tasks in order to achieve the optimal strategy selection problem of cooperation and competition mode. The optimal strategy selection problem is analyzed and compared through the design of different experimental scenarios under the multi-agent cooperative competition and multi-type combination mode.

2. Agent Strategy Algorithm

2.1 Markov Model

The Markov model for multi-agents consists of the following parts:

(1) A group of states S , Used to describe the possible states of all agents.

(2) A group of agent behaviors $A_1, A_2 \cdots A_N$.

(3) A set of observations for each agent $O_1, O_2 \cdots O_N$.

In the experiment, each agent uses a random strategy $\pi_{\theta i}: O_i \times A \mapsto [0,1]$ and the agent's own state transition function $T: S \times A_1 \times A_2 \times \cdots \times A_N \mapsto S$ to select each agent i Action. Each agent i receives a reward according to its own state and the action $r_i: S \times A_i \mapsto \mathbb{R}$ selected by the agent, and receives the state of the private observation environment related to the state $o_i: S \mapsto O_i$. The initial state of the agent is determined by $\rho: S \mapsto [0,1]$. The goal of each agent is to obtain the maximum expected reward as in formula (1):

$$R_i = \sum_{t=0}^{T} \gamma^t r_i^t \tag{1}$$

In the formula: γ indicates the discount factor, and T indicates the time range.

2.2 Actor-Critical Framework Algorithm Principle

The Actor-Critic (AC) algorithm is divided into two parts, the Actor network and the Critic network. The Actor network is responsible for selecting the appropriate action in the continuous action space. Critic network is responsible for assessing the performance of the Actor network and guiding the Actor's next stage of action, while implementing single-step updates. The Actor network will usually adopt the strategy gradient method, and its parameter gradient is:

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n-1}^{N} \sum_{t=1}^{T_n} Q^{\pi\theta}\left(s_t^n, a_t^n\right) \nabla \log p_\theta\left(a_t^n \mid s_t^n\right) \tag{2}$$

The Critic network updates its parameters according to the squared error between the estimated Q value and the actual Q value, and its loss function is:

$$loss = \frac{1}{N} \sum_{n-1}^{N} \sum_{t=1}^{T_n} \left(r_t^n + \max_{a_{t+1}^n} Q^{\pi\theta}\left(s_{t+1}^n, a_{t+1}^n\right) - Q^{\pi\theta}\left(s_t^n, a_t^n\right) \right)^2 \tag{3}$$

Fig. 1 is a schematic diagram of the overall framework of the Actor-Critic algorithm.

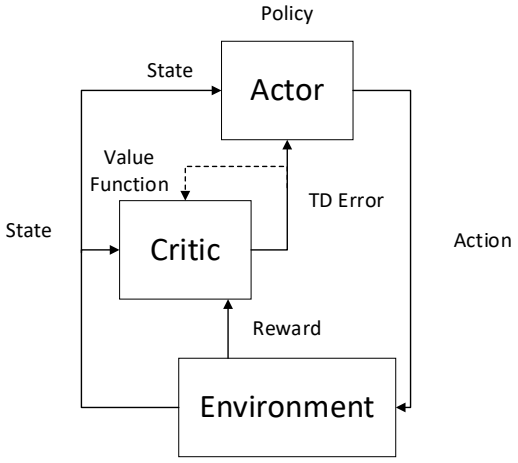

Fig. 1 AC algorithm framework

3. Improved multi-agent strategy algorithm

3.1 Multi-agent strategy model

The article uses deep reinforcement learning as the basis for agent strategy learning. The overall environment setting conforms to the Markov model, and an extended strategy gradient method based on actor-critic framework is proposed. Agent training uses an independent execution framework for centralized strategy learning. The algorithm expands the strategy evaluation information based on the AC framework, assuming that there are N agents, and the agent's policy parameter is

θ=$\{\theta_1,\cdots,\theta_N\}$. The policy set of all agents is π=$\{\pi_1,\cdots,\pi_N\}$. So the expected gradient of agent i according to $J(\theta_i)=E[R_i]$ is:

$$\nabla_{\theta_i} J(\theta_i)=E_{s\sim p^{\mu},a_i\sim\pi_i}[\nabla_{\theta_i}\log\pi_i(a_i\mid o_i)\mathrm{Q}_i^{\pi}(s\_,a_1,\cdots,a_N)] \tag{4}$$

In the formula, $\mathrm{Q}_i^{\pi}(x,a_1,\cdots,a_N)$ represents a centralized agent action value function. It takes the actions of all agents $a_1,\cdots,a_N$ as input and outputs state x and Q value of agent i. In the simplest case, $s\_=(o_1,\cdots,o_N)$ contains the environment observed by all agents, and can also contain more information. Q_i^{π} for each agent is learned independently, so each agent can have any reward structure, including conflict rewards in the competitive environment of the agent.
Extending the above formula (4) to the agent's deterministic strategy method. Assuming that N consecutive policies are $\mu_{\theta i}$ and the parameter θ_i is recorded as μ_i, then its gradient can be expressed as:

$$\nabla_{\theta_i} J(\mu_i)=E_{x,a\sim D}[\nabla_{\theta_i}\mu_i(a_i\mid o_i)\nabla_{a_i}\mathrm{Q}_i^{\mu}(s,a_1,\cdots,a_N)\mid_{a_i=\mu_i(o_i)}] \tag{5}$$

In the formula, experience pool D contains $(x,x',a_1,\cdots,a_N,r_1,\cdots,r_N)$ information, which records the experience of all agents. The centralized agent action value function Q_i^{μ} can be described as:

$$Loss(\theta_i)=E_{s,a,r,s'}[(\mathrm{Q}_i^{\mu}(s,a_1,\cdots,a_N)\text{-}y)^2],y=r_i+\gamma\mathrm{Q}_i^{\mu'}(s',a_1',\cdots,a_N')\mid_{a_j'=\mu_j'(o_j)} \tag{6}$$

In the formula, μ'=$\{\mu_{\theta_1'},\cdots,\ \mu_{\theta_N'}\}$ is an agent strategy set with a delay parameter of θ_i'. The above method is the basic idea of an improved multi-agent strategy algorithm-if the actions of all agents are known, the environment is stationary even if the strategy changes. Because:

$$P(s'\mid s,a_1,\cdots,a_N,\pi_1,\cdots,\pi_N)=P(s'\mid s,a_1,\cdots,a_N)=P(s'\mid s,a_1,\cdots,a_N,\pi_1',\cdots,\pi_N') \tag{7}$$

This holds for any agent policy set $\pi_i\neq\pi_i'$.

3.2 Agent strategy inference

In order to avoid assuming the strategies of other agents, the real strategy μ_j of each agent i to agent j can be approximated by $\hat{\mu}_{\varnothing_i^j}$ ($\varnothing$ is an approximate parameter, hereafter referred to as $\hat{\mu}_i^j$). The entropy regularizer can be used to learn the agent j strategy:

$$L(\varnothing_i^j)=-E_{o_j,a_j}[\log\hat{\mu}_i^j(a_j\mid o_j)+\lambda H(\hat{\mu}_i^j)] \tag{8}$$

In the formula, H is the entropy of j agent's strategy. According to the approximate calculation of the strategy, the formula can be obtained:

$$\hat{y}=r_i+\gamma\mathrm{Q}_i^{\mu'}(s',\hat{\mu}_i'^1(o_1),\cdots,\mu_i'(o_i),\cdots,\hat{\mu}_i'^N(o_N)) \tag{9}$$

In the formula, $\hat{\mu}_i'^j$ is the approximate value of agent strategy $\hat{\mu}_i^j$. Before updating the agent's Q function Q_i^{μ}, we use a single gradient step size of each agent j in the experience pool to update $\varnothing_i^j$.

3.3 Agent Cooperation

In the interaction between multi-agent reinforcement learning and the environment, there is a problem of environmental instability caused by policy changes, which affects decision-making. Therefore, in order to reduce the impact of this situation, multi-agent training can develop strategies by evaluating the behavior of the other party. We can train K different coping strategies to randomly select a specific sub-strategy for each agent to be executed in each round, in order to train the agents to evaluate each other's strategy-making methods.

Assuming that μ_i is a set of K different strategies for the agent, and the sub-strategy k is represented by $\mu_i^{(k)}$, the formula for the maximum value of each agent i is:

$$M_e(\mu_i) = E_{k\sim unif(1,K),s\sim p^{\mu},a\sim\mu_i^{(k)}}[R_i(s,a)] \tag{10}$$

Since the agent will execute different sub-strategies in different scenarios, an experience pool $D_i^{(k)}$ is maintained for each sub-strategy $\mu_i^{(k)}$ of agent i. Then the gradient of $\theta_i^{(k)}$ can be expressed as:

$$\nabla_{\theta_i^{(k)}} J_e(\mu_i) = \frac{1}{K} E_{X,a\sim D_i^{(k)}}[\nabla_{\theta_i^{(k)}} \mu_i^{(k)}(a_i \mid o_i) \nabla_{a_i} Q^{\mu_i}(x, a_1, \cdots, a_N)|_{ai=\mu_i^{(k)}(o_i)}] \tag{11}$$

4. Experimental result

The parameters of the improved multi-agent strategy method are composed of two layers of ReLU and MLP. The Gumbel-Softmax evaluator is used to better communicate discrete messages between agents. The experiment evaluates the state and rewards of the improved multi-agent strategy method in the case of cooperative and hostile environment.

The state of the improved multi-agent strategy algorithm in the cooperative task environment of the agent is shown in Fig. 2. The goal of the experiment in the cooperative environment is that the agent is as close as possible to the target under the guidance of the commander. The red dot in the figure indicates the target position, and the green dot is the commander.

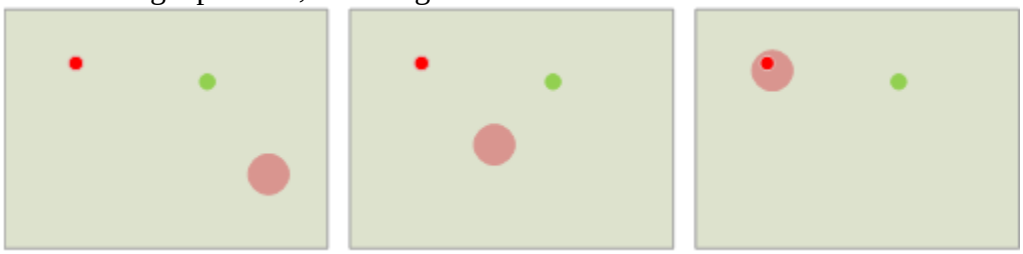

Fig. 2 Cooperative environment movement of the algorithm in this paper

After training 30,000 times in the cooperative task environment of the agent, we calculate the average success rate of the agent reaching the target in each round, and record the average distance of the cooperative agent from the target. Table 1 is the statistical comparison data between the improved algorithm of this paper and AC algorithm, depth strategy gradient algorithm, etc.

Table 1 Reach rate and distance in cooperative agent tasks

Policy	Reach rate	Distance
ACEM(our)	81.8%	0.247
DDPG	36.2%	0.439
DQN	28.7%	0.712
AC	20.5%	2.364

The objective of the experiment in the hostile environment is to avoid the collision of the red agent with the black agent as much as possible. The state of the improved multi-agent strategy algorithm in the hostile environment of the agent is shown in Fig. 3 below.

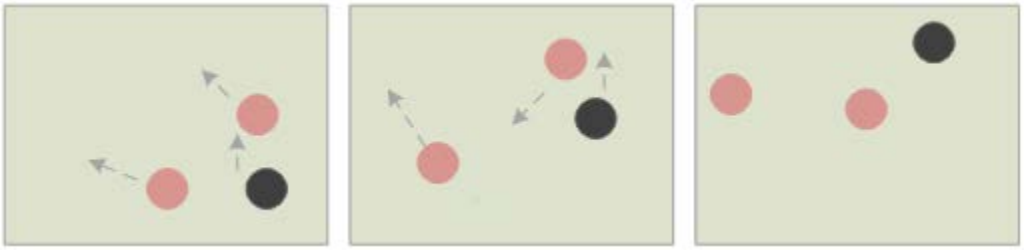

Fig. 3 Hostile environment movement of the algorithm in this paper

5. Conclusions

This paper introduces an extended multi-agent strategy gradient method based on actor-critic (ACEM). It uses extended strategy evaluation information to perform reward calculation after agents perform strategic actions, and trains a series of coping strategies by inferring other agent strategies. In each round, the agent randomly selects sub-strategies to reduce the impact of environmental instability on strategy selection. From the experimental results, the agent can choose effective strategies in a cooperative and hostile environment. However, this method requires higher hardware equipment, and the convergence speed is slower when the number of agents is large, so the further improvement is needed.

References

[1] C.Altafini.Consensus problems on networks with antagonistic interactions [J], IEEE Trans. on Automatic Control, 2013, 58 (4): 935-946.

[2] J. Hu, Z. H. Xiao, Y. L. Zhou,et al.Formation control over antagonistic networks [C]. Proceeding of 32nd Chinese Control Conference, July 26-28, Xi'an, China, 2013, : 6879-6884.

[3] E.V., Maria, P.Misra.On the consensus and bipartite consensus in high-order multi-agent dynamical systems with antagonistic interactions, Systems & Control Letters, 2014, 66:94-103.

[4] Z.Meng, G. Shi, K. H. Johansson, et al. Modulus Consensus over Networks with Antagonistic Interactions and Switching Topologies.

[5] Q.Wang, Y. Wang. Cluster synchronization of a class of multi-agent systems with a bipartite graph topology. [J]Science China Information Sciences January, 2014, 57(1): 1-11.

[6] Qi G J, Larochelle H, Huet B, et al. Guest Editorial: Deep Learning for Multimedia Computing [J]. IEEE Transactions on Multimedia, 2015, 17 (11): 1873-1874.

[7] El-Tantawy S, Abdulhai B, Abdelgawad H. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto [J]. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(3): 1140-1150.

[8] Qiu J, Gao H, Ding S X. Recent advances on fuzzy-model-based nonlinear networked control systems: a survey [J]. IEEE Transactions on Industrial Electronics, 2016, 63 (2): 1207-1217.

[9] Huang S C, Do B H. Radial basis function based neural network for motion detection in dynamic scenes [J]. IEEE transactions on cybernetics, 2014, 44(1): 114-125.