Adaptive learning and evolution framework of intelligent combat behavior modeling based on EXIT framework reasoning

Jun Lai 1,a, Xi-Liang Chen2,* Zhi-Xiong Xu3,b

1 Army Engineering University of PLA, Nanjing, China
2 Army Engineering University of PLA, Nanjing, China
3 Army Engineering University of PLA, Nanjing, China

*Corresponding author e-mail: 383618393@qq.com
a2568754202@qq.com, b xu.nj@Foxmail.com

Abstract. The intelligent combat behavior modelling has always been a hot topic in the field of artificial intelligence and military. Aiming at the requirement of development of intelligent equipment in complex battlefield, this paper proposes an adaptive learning and evolution framework of intelligent combat behavior modeling based on EXIT framework reasoning, and provides theoretical, technical and methodological support for more efficient implementation of the evaluation and application verification of intelligent action plan.

1. Introduction

The essence of AI is to simulate the way of human's thinking to solve problems. When people makes complex decision-making activities, they do not just rely on knowledge or intuition to make decisions, but make final decisions through multi-step planning and reasoning. Reinforcement learning can be regarded as the process of forming intuitive model through the interaction between the agent and the environment. Combining it with reasoning model is an important way to improve learning and decision-making ability. How to take advantage of artificial intelligence technology to construct intelligent combat behavior modelling to make the machine reach or approach the level of cognition and decision-making of excellent military personnel is an important theoretical and technical problem faced by intelligent command and decision-making research.

Alpha Go Zero, which combines the Monte Carlo tree search, reinforcement learning method and the Expert Iteration (EXIT) framework based on Dual process theory, gives us inspiration to imitate the process of human’s thinking. In this paper, a deep reinforcement learning method based on reasoning model is studied. A game reasoning model is added to reinforcement learning to enable agents to learn in self-play.

2. Deep reinforcement learning framework based on reasoning model

Commanders can process information much faster than computers, but their thinking activities of decision-making and reasoning are the key to victory in complex warfare. Therefore, when constructing the intelligent decision-making model, we not only need to learn the process of the commander accumulating experience, but also need to simulate the reasoning mechanism of the commander. Reinforcement learning can be regarded as the process of interaction between the agent
and the environment to form intuitive model. Combining it with reasoning model is an important way to improve learning and decision-making ability.

At the same time, we note that reinforcement learning can only learn one agent's decision-making policy, and the opponent's policy must be given by the environment. In this case, the quality of the policy learned by the agent largely depends on the quality of the opponent's policy. When the quality of opponent's policy is not high or relatively fixed, the generalization ability of the policy model learnt by agent is weak in practical application. The game reasoning model is used to model the game process of both sides, and adding the game reasoning structure to reinforcement learning can realize the self-game of reinforcement learning and improve the generalization ability of intelligent decision model.

Based on the above analysis, this paper proposes a deep reinforcement learning framework based on reasoning model on the basis of summarizing the research work of relevant scholars. The intelligent decision process is divided into off-line training stage and on-line deployment stage. According to the different types of models used in the two stage, they are divided into two technical solutions, as shown in Figure 1. The deep reinforcement learning method based on reasoning model is an example of the second schemes.

Scheme 1: In the off-line training stage, the machine learning method is used to learn the existing command decision data, and the intuitive model is obtained. In the online stage, the intuitive model and the reasoning model are deployed synchronously. The intuitive model outputs the possible actions and corresponding probabilities of the situation. The reasoning model performs iterative reasoning according to the results given by the intuitive model and outputs the final actions.

Scheme 2: In the off-line training stage, intuitive model and reasoning model are used to guide the action selection of the reasoning model, and the data generated by the reasoning model is used to upgrade the intuitive model. After repeated iterations, we finally export the trained intuition model. In the online deployment stage, we directly use the trained intuition model to make decisions.

**Figure 1.** Adaptive learning and evolution of online / offline intelligent combat behavior model based on reasoning model

3. **Reasoning model and deep reinforcement learning method based on reasoning model**

Reinforcement learning can be regarded as the process of forming intuitive model through the interaction between the agent and the environment. Combining it with reasoning model is an important
way to improve learning and decision-making ability. Based on the Expert Iteration (EXIT) framework of Dual Processing Theory (Dual Processing Theory), complex decision-making is carried out by combining Monte Carlo Tree Search (MCTS) and Reinforcement Learning (RL). In the training phase, MCTS is used as the inference model, and the decision-making ability is improved step by step through the closed learning loop.

This paper intends to use the generalized policy iteration framework in reinforcement learning to learn. The idea of generalized policy iteration is to learn the optimal policy by alternating policy evaluation and policy improvement during the learning process, as shown in Figure 11. Policy evaluation, through numerical iteration algorithm to constantly calculate and update the value function of the current policy under each state; policy improvement, using the new value function to build action policy, continue to interact with the environment. Policy evaluation and policy improvement cycle iteration will eventually get the best policy. Policy evaluation and policy improvement are completed by MCTS algorithm. Based on the reasoning model, MCTS is introduced into reinforcement learning as a game reasoning structure. The policy is improved by MCTS action selection and evaluated by node value estimation of MCTS.

![Figure 2. Generalized policy iteration](image)

The whole training process consists of two key stages: the self playing stage and the neural network training stage. In each round of self-play, MCTS gets policy $\pi$ and result $z$ by searching. The policy $\pi$ of MCTS can be regarded as the improvement of neural network policy $p$, and the result $z$ is the evaluation of current policy. In the stage of self playing, MCTS algorithm simultaneously improves policy and policy. In the training phase of the neural network, the policy function and the evaluation function tend to the policy and the evaluation of MCTS, $p \rightarrow \pi$ and $v \rightarrow z$, respectively. Say concretely:

In the self game stage, we use the MCTS algorithm to save 4 values for each side $(s, a)$:

$$\{N(s, a), W(s, a), Q(s, a), P(s, a)\}$$

The MCTS was merged from four stages into three stages, namely, the number of visits, the cumulative action value, the average action value and the prior probability of the action.

1. Selection stage: According to the Monte Carlo tree $a = argmax(Q(s, a) + u(s, a))$ with prior knowledge, action selection is carried out, and noise $\eta$ is introduced into the prior probability. (Obeying Dirchlet (0.03) distribution)

$$P(s, a) = (1 - \epsilon)P(s, a) + \epsilon\eta$$  \hspace{1cm} (1)

Among them, $\epsilon$ is inertia factor.

Starting from the root node $s_0$, each step is selected according to the Monte Carlo tree with prior knowledge in the process of reaching the leaf node $s_L$ through $L$ steps.

2. Extension phase and evaluation phase: After arriving at the leaf node $s_L$, the probability value $p$ and evaluation value $v$ of the current node are obtained by calling the neural network $(p, v) = f_{\theta}(s_L)$. The initialization node is:
{N(s_L, a) = 0, W(s_L, a) = 0, Q(s_L, a) = 0, P(s_L, a) = p_0} \tag{2}

3. Return phase: After the extension phase and evaluation phase, the latest information from the leaf node is returned to the root node:

\begin{align*}
N(s_t, a_t) &+= 1 \tag{3} \\
W(s_t, a_t) &+= ν \tag{4} \\
Q(s_t, a_t) &= \frac{W(s_t, a_t)}{N(s_t, a_t)} \tag{5}
\end{align*}

4. Actual execution: Back to the root node, the actual action is determined according to the following policies:

\[ \pi(a|s_0) = \frac{N(s_0, a)^{1/τ}}{\sum_b N(s_0, b)^{1/τ}} \tag{6} \]

Among them, \( τ \) is the simulated annealing parameter, which is used to control the extent of exploration.

After the execution of the actual action, the information of each sub node of the search tree extension is preserved. In the next step, the new node is taken as the root node to continue the MCTS search.

5. At the end of a round, the result \( z \) is obtained, and the battlefield situation and actions of each step are recorded as \( (s_t, π_t, z_t) \), in which \( z_t \) represents the result from the perspective of the current agent. The data \( (s_t, π_t, z_t) \) of each step is stored in the memory unit \( D \).

After the end of each round, the training phase of the neural network is started. A neural network is used to simultaneously learn the probability distribution of each action in the current state and the estimation of the overall situation. \( (p, ν) = f_θ(s) \). Sampling data \( (s, π, z) \) from memory units \( D \) is trained to make the policy function and the valuation function tend to the policy and the valuation of MCTS, \( p → π \) and \( ν → z \). Therefore, the objective function constructed is

\[ \text{loss} = (z - ν)^2 - π^T \log p + c\|θ\|^2 \tag{7} \]

After the loss function as shown in Eq. (7) is obtained, the policy network and the value network can be trained according to the loss function, which are the basis of the behavior of the intelligent combat entity. In the pre-war planning stage, tactical action planning can be carried out according to the strategy network and value network, and the operational action sequences of each operational entity can be generated, which can assist military personnel and autonomous entities in making operational plans. In the combat implementation stage, the decision-making can be adjusted according to the real-time situation of the battlefield and the changes of combat resources, assisting military personnel and autonomous entities to make decision-making in the near future, and dynamically adjusting the operations of combat units. Its core function is to construct an intelligent decision-making model of operational entities, to realize action selection under complex and changeable battlefield situation, to provide support for the generation and optimization of operational sequences in the pre-war planning stage, and to support the commander's decision-making in the combat implementation stage.

4. Deep reinforcement learning test based on MCTS

In order to validate the effectiveness of the algorithm in the Tactical Simulation platform, a deep reinforcement learning experiment based on MCTS was carried out.

The specific parameters for the experiment are as follows: The state is represented as a 12 dimensional vector. \( s = \{x_1, y_1, h_1, m_1, ad_1, f_1, x_2, y_2, h_2, m_2, ad_2, f_2\} \). Among them, \( x \) and \( y \) represent the position of each combat unit (horizontal and vertical coordinates); \( h \) represents the number of remaining tanks in each combat unit; \( m \) represents the aerodynamic value of each combat unit; \( ad \) represents the attacker or defender, the defender \( ad = 0 \), the attacker \( ad = 1 \); \( f \) represents whether each combat unit occupies or not. Seize control points, \( f = 0 \) represents unoccupied, \( f = 1 \) represents occupied. The neural network of the policy model is always trained.
from the first perspective, so it is necessary to transform each state into the current actor’s perspective, that is, \( \{x_1, y_1, h_1, m_1, ad_1, f_1\} \) is the current actor’s state description vector, and \( \{x_2, y_2, h_2, m_2, ad_2, f_2\} \) is the other’s state description vector; the number of actions per combat unit is still 18. Set the parameters \( c_{\text{puct}} = 5 \), simulated annealing parameter \( \tau = 1 \) and inertia factor \( \epsilon = 0.25 \) in MCTS, the memory unit \( D \) of self-playing data is \( \text{size} = 10^4 \), and the new data replaces the old data when the game data exceeds; train the neural network once after each game round, and train the sampling size \( \text{batch size} = 512 \); the neural network structure is shown in Figure 3. shows that the learning rate of the neural network is \( \text{learning rate} = 2 \times 10^{-3} \).

**Figure 3.** The structure of neural network

In order to reduce the amount of calculation and save training time, this paper introduces a simple combat rule to guide the selection of nodes in MCTS in the early stage of self-play, and realizes the "hot start" of self-play.

In the course of training, the neural network was tested 10 times every 100 rounds. The neural network was chosen as the red-square intelligence and the artificial defense rules as the blue-square intelligence.

The variation curves of loss function and entropy of policy network are shown in Figure 4 and Figure 5.

**Figure 4.** The change curve of loss function in training process

**Figure 5.** Entropy change curve of policy in training process

In the course of the game training, the neural network was tested 10 times every 100 rounds. The red side used the neural network which was trained at present as the attack side, the blue side used the artificial rules as the defense side, and the number of winners is shown in Table 1. From the results, we can see that the Red Square's success rate of artificial rules is gradually increasing with the training process, but it can still beat the Blue Square with a winning rate of about 60% under the circumstance that the learning is not completely focused on the Blue Square's reinforcement training, which shows that the deep reinforcement learning method based on reasoning model has a strong generalization ability.
Table 1. The winning number of the red side tested in the training process 10 times.

| Rounds | Number | Rounds | Number | Rounds | Number | Rounds | Number |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 100    | 0      | 1100   | 3      | 2100   | 2      | 3100   | 2      | 4100   | 7      |
| 200    | 0      | 1200   | 2      | 2200   | 3      | 3200   | 3      | 4200   | 6      |
| 300    | 2      | 1300   | 1      | 2300   | 4      | 3300   | 5      | 4300   | 5      |
| 400    | 1      | 1400   | 2      | 2400   | 2      | 3400   | 2      | 4400   | 6      |
| 500    | 1      | 1500   | 2      | 2500   | 3      | 3500   | 3      | 4500   | 4      |
| 600    | 3      | 1600   | 1      | 2600   | 1      | 3600   | 4      | 4600   | 5      |
| 700    | 2      | 1700   | 3      | 2700   | 4      | 3700   | 5      | 4700   | 6      |
| 800    | 1      | 1800   | 3      | 2800   | 3      | 3800   | 6      | 4800   | 5      |
| 900    | 2      | 1900   | 4      | 2900   | 5      | 3900   | 7      | 4900   | 7      |
| 1000   | 3      | 2000   | 3      | 3000   | 6      | 4000   | 8      | 5000   | 6      |

To sum up, deep reinforcement learning method based on reasoning model has achieved good results in tactical simulation tasks, and has a strong applicability. The learned policy generalization ability is strong, online game time-consuming is relatively short.

5. Conclusion

The training process of reinforcement learning is the proceeds of forming intuitive model during the interaction between the agent and the environment. Combining reinforcement learning with reasoning model is an important way to improve learning and decision-making ability. In this paper, deep reinforcement learning method based on reasoning model is studied, and a framework of deep reinforcement learning based on reasoning model is proposed. The paper carries on the experiment in the tactical simulation platform, the experimental results show that both red and blue sides have learned the effective attack and defense policy through the self-game, and the decision-making model has stronger generalization ability. The future work should focus on the reasoning model and the algorithm research under the condition of incomplete information combat.

References

[1] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587):484-489.
[2] GUO X, SINGH S, LEE H, et al. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning [C]// Proc. of the Neural Information Processing Systems. MA: MIT Press, 2014: 3338-3346.
[3] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676):354-359.
[4] Silver D, Hubert T, Schrittwieser J, et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm [EB/OL]. [2018-2-2]. https://arxiv.org/pdf/1712.01815.pdf.
[5] Anthony T, Tian Z, Barber D. Thinking Fast and Slow with Deep Learning and Tree Search[J]. NIPS 2017.
[6] Browne C B, Powley E, Whitehouse D, et al. A Survey of Monte Carlo Tree Search Methods[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2012, 4(1):1-43.
[7] Auer P, Cesa-Bianchi N, Fischer P. Finite-time Analysis of the Multiarmed Bandit Problem[J]. Machine Learning, 2002, 47(2-3):235-256.
[8] Kocsis L, Szepesvári C. Bandit based monte-carlo planning[J]. Lecture Notes in Computer Science, 2006, 4212:282-293.
[9] Yoon S, Fern A, Givan R, et al. Probabilistic planning via determinization in hindsight[C]// National Conference on Artificial Intelligence. AAAI Press, 2008:1010-1016.
[10] Borsboom J, Saito J T, Chaslot G, et al. A Comparison of Monte-Carlo Methods for Phantom Go[J]. 2008.

[11] Bjarnason R, Fern A, Tadepalli P. Lower bounding Klondike Solitaire with Monte-Carlo planning[C]// Nineteenth International Conference on International Conference on Automated Planning and Scheduling. AAAI Press, 2009:26-33.