Play games using Reinforcement Learning and Artificial Neural Networks with Experience Replay

Meng Xu
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
menghsu@mail.nwpu.edu.cn

Haobin Shi*
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
shihaobin@nwpu.edu.cn

Yao Wang
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
biu_vip@163.com

***Abstract*—Reinforcement learning is a self-learning algorithm in which an agent acquires experience through continuous interaction with the environment, which is more like the process of human or animal learning. Reinforcement learning is widely used in the field of playing games, and the classical reinforcement learning algorithm can easily produce Curse of Dimensionality when the state dimension is too large. In order to improve the convergence rate of reinforcement learning, a method of training Non Player Character (NPC) in games using Sarsa learning algorithm is proposed. The artificial neural network is used to approximate the value function. In order to make better use of experience, this paper sets up double neural networks, and uses experience memory to store experience, and uses experience replay to speed up the convergence of sarsa learning. Using the method presented in this paper to train NPC, we can find the NPC which is trained by the method has more learning ability than the classical reinforcement learning.**

Keywords—reinforcement learning, ***artificial neural network***, ***experience replay***, ***Non Player Character***

I. INTRODUCTION

Artificial Intelligence has been widely used in the field of game development. Artificial Intelligence can develop more intelligent characters. How to use artificial intelligence to train the NPC with stronger learning ability has attracted the attention of many scholars. Reinforcement learning is a way to make intelligent decisions by acquiring experience, which has rapidly become one of the most popular AI methods in recent years.

Reinforcement learning has been applied in many research fields. Scholars in the field of reinforcement learning also carried out a lot of research, which in the field of game development also has a lot of research cases[1-3]. In literature[4], Q-learning algorithms are used in computer video games, and researchers use Q-learning algorithms to train NPCs to learn and fight. But the classical Q-learning algorithm is less convergent and less stable in specific application scenarios. The scholar proposed to use the SARSA algorithm[5], But classical reinforcement learning has no way to deal with continuous state space learning problems. Artificial neural network has strong generalization ability, so some scholars propose to use artificial neural network to study with reinforcement learning[6],and this method does not make full use of the experience. In this paper, double neural networks is proposed to approximate the Q value. In order to make better use of the past experience, this paper uses the experience replay and sets up the double neural network: one is the current value network and the other is the target value network.

II. SARSA LEARNING ALGORITHM WITH EXPERIENTIAL REPLAY

A. Reinforcement learning

Markov decision-making process is a continuous transition process of state, in which the current state and the next state is not associated, as shown in Figure 1. Markov decision-making process is used to solve the problem of control or decision. Markov decision-making process can be described with the following factors: state, action, reward, state transition probability, and discount factor. The purpose of Reinforcement Learning is to obtain the maximum cumulative reward value under Markov process, and the cumulative reward can be expressed using the state-action value function, which is described by the Bellean equation:

$$V_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^p R_{t+p+1} \mid s_t = s, a_t = a] \tag{1}$$

The optimal state-action $Q^*(s,a)$ value function is the state-action Value function with the highest value in all policies.

$$\begin{cases} V^*(s,a) = \max\limits_{\pi} V_\pi(s,a) \\ V^*(s,a) = \sum\limits_{s' \in S} P(s,a,s')[R + \gamma \max\limits_{a' \in A} V^*(s',a')] \end{cases} \tag{2}$$

Temporal-Difference learning is a model-free reinforcement learning method for solving the series state decision. Because of no model, the whole successor state of the current state cannot be obtained, and the learning agent obtains the next state of the current state through the sampling method. Using the thought of Temporal-Difference learning, this paper rewrites the formula of the state-action value

$$V(s_t,a_t) = V(s_t,a_t) + \alpha(R + \gamma \max_{a_{t+1} \in A} V(s_{t+1},a_{t+1}) - V(s_t,a_t)] \tag{3}$$

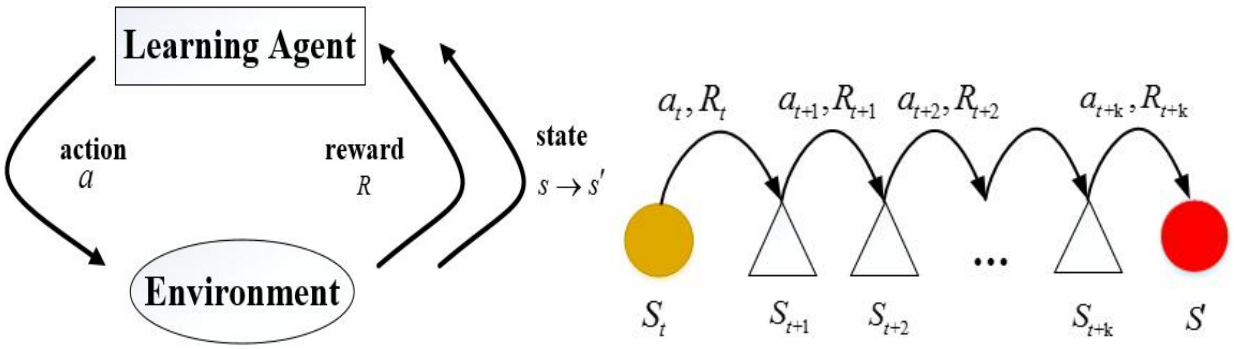
If the state-action value function uses the greedy strategy to select the action, then it becomes the Q-Learning algorithm,

- Research work is supported by Aeronautical Science Fund (No.2016ZC53022) , National key research and development program of China (No.SQ2017YFGX060091) and The 2018 Graduate Starting Seed Fund of Northwestern Polytechnical University(No.ZZ2018169).

- Haobin Shi is the corresponding author.

978-1-5386-5892-5/18/$31.00 ©2018 IEEE
ICIS 2018, June 6-8, 2018, Singapore

and the Q-Learning algorithm is a model-agnostic reinforcement learning algorithm proposed by Watkins[7].



（a）Reinforcement learning (b) Markov decision-making process

Fig.1 Markov Decision Process and Reinforcement Learning

B. SARSA learning algorithm using experience replay

Sarsa learning is an on-policy learning algorithm, the iterative of which for the value function is the method of strict temporal-difference learning. Sarsa learning differs from Q learning, and the Sarsa learning still uses greedy strategy to select action in the next state[8]. But in some control problems, Sarsa learning algorithm has faster convergence rate than Q-learning algorithm. The Sarsa learning algorithm chooses the action according to the current state s_t , and uses $\varepsilon - greedy$ algorithm to select action p_t . After action p_t is performed, the current state s_t moves to next state s_{t+1} ,and gets rewards R . When agent in state s_{t+1} , uses the $\varepsilon - greedy$ algorithm to select next action p_{t+1} . Finally, this paper updates the value function $Q(s_t, p_t)$ according to the formula.

$$Q(s_t, p_t) = Q(s_t, p_t) + \alpha[R + \gamma Q(s_{t+1}, p_{t+1}) - Q(s_t, p_t)] \quad (4)$$

This paper sets $T_t = R + \gamma Q(s_{t+1}, p_{t+1})$ to target Q value, and set $D_t = Q(s_t, p_t)$ to current Q value, and set $\Delta_t = T_t - D_t = R + \gamma Q(s_{t+1}, p_{t+1}) - Q(s_t, p_t)$ to error.

In this paper, we propose to set up the experiential memory, store the four-tuple $\langle S, P, S', R \rangle$ which is used to describe the experience, and update the value function through experience replay to speed up the convergence of Sarsa learning. $M(s_t, p_t, R, p_{t+1})$ is stored in the experience memory after the next action p_{t+1} is obtained. The Sarsa algorithm with experience replay is shown in Figure 4.

III. Reinforcement Learning Based on Neural Network with Experience Replay

A. Double artificial neural network with experience replay

Artificial neural network is a kind of computing model which is connected by many neurons, the neural network used in this paper has three layers structure, the first layer is the input layer, the second layer is hidden layer and the third layer is the output layer. In artificial neural network, the most commonly used is the Back Propagation neural network, and this paper is using the Back Propagation neural network. Back propagation can reduce the amount of computation. The classical reinforcement learning algorithm is to store the value function in the form of table, but this method cannot solve the large-scale continuous state space problem, and the neural network has strong generalization ability, and it can approximate the value function $Q(s, a, w)$ in Sarsa learning. ω represents the parameters of neural network, the neural network updates the weights by the way of Back propagation, and finally minimizes the loss function $L(\omega)$.

$$\begin{cases} L_t(\omega) = E\left[(T_t - Q(s_t, p_t; \omega_t))^2\right] \\ T_t = R + \gamma Q(s_{t+1}, p_{t+1}, \omega^*) \end{cases} \quad (5)$$

ω^* is the weight value of the target neural network, and neural network has the target neural network and the current neural network. The two neural networks have exactly the same three-layer structure, and the parameters of the current neural network ω_t are updated by Back propagation. The parameters of the target neural network ω^* are updated by the parameters of the current neural network each τ time steps, and the environment model is stored in four-tuple $S(s_t, p_t, r_t, p_{t+1})$, and the environment model is saved in the experience memory.

B. Sarsa learning based on double artificial neural network with experience replay

The basic framework of the method presented in this paper is shown in Figure 3. At the beginning of the round, when the NPC is in state s_t , put the s_t input into the current neural network and output is $Q(s_t, p_t; \omega_t)$ .Selecting an action p_t based on the $\varepsilon - greedy$ algorithm, the state s_t moves to the next state s_{t+1} , and environment return rewards r_t , and store four tuples $S(s_t, p_t, s_{t+1}, r_t)$ in experience memory. The experiential memory inputs s_{t+1} into the target neural network

and then gets the value function $Q(s_{t+1}, p_{t+1}, \omega^*)$ based on the $\varepsilon - greedy$ algorithm, and Calculate Loss function $L_t(\omega)$.

Then the weights of the current neural network are updated using the Back propagation, which is recorded as a current neural network parameter updating process. The experience memory then randomly selects a previous state s_t', and it randomly selects actions a_t' in that state, and generates the next state s_{t+1}' and reward r' .Updating the current neural network parameters according to a current neural network parameter updating process, which repeats N times, and a complete current neural network parameters updating process is performed. After τ times of the complete current neural network parameter updating process, the parameters of the current neural network ω_t are copied to the target neural network parameters ω^*.

IV. Experiment and analysis

A. Experimental preparation

The experiment in this paper is carried out under the Robocode platform[9], where two or more tank robots fight for victory. Robocode is a 1000*800 pixel two-dimensional simulation battle platform, and the tank itself size is 36*45 pixel, under the platform to place two of robotic tanks. The robotic tank named Firebot is designed under the Robocode platform, and the motion strategy of Firebot is set as follows: it keeps a random movement and shoots at each place after the target is found. Using three different learning Algorithms: the Classical Sarsa learning Method (SARSA), the Sarsa learning Method with the double neural network (DB-SARSA), and the Sarsa learning method based on double the neural network with the experience replay(RDB-SARSA), to train tank Rlbot and fight with Firebot, and tank Rlbot, the NPC, is the protagonist of this paper. The experimental platform used in this paper is shown in Figure 2. The two sides at the beginning of each round have 100 points of life. When one of the life point is 0, the round is end. Every 40 rounds recorded as a session, the experiment was conducted 50 sessions. The reinforcement learning model is based on the Markov decision process. Generally speaking, the construction of the reinforcement learning model needs the state space, the action space, the reward function, and the reinforcement learning model for the Battle of the Tank NPC is described as follows.

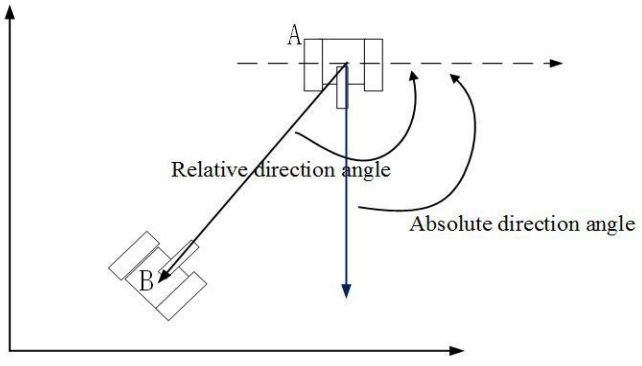
State space: The state information of the tank NPC in the battlefield is described by the absolute direction angle, the relative direction angle, the distance between the tank NPC, Whether the tank NPC collided with the wall, and whether or not hit by the bullet, the state space is the combination of these five states. The absolute direction angle range is $0 \sim 360^\circ$. The absolute direction angle is discretized into four kinds of states. The same relative direction angle is also discretized into four states,and the distance between the tanks is defined as the distance between the tank NPC and the enemy's center point. According to the size of the tank NPC itself and the battlefield, the distance is divided into 20 discrete values, and each discrete value is a multiple of 30 pixels.The definition of value 0 means that the tank NPC did not collide to the wall, using a value of 1 means that the NPC hit the wall, the same, using a value of 0 means that the NPC was not hit by a bullet,and using a value of 1 for the NPC was shot.

Action space: There are two basic movement modes of the tank NPC: Moving and Rotating. This paper defines six different movements: the forward, backward, to the left front, to the right front, to the left back, and to the right back as action space.

Reward function: In the game, when the tank NPC hits the wall, hits the other side, or shoots the enemy and so on, the life point of tank NPC will change. Different state has the different life point change rule, and tank NPC firing bullets need to consume their own life points, and different bullets have different energy. The function getpower () can obtain the bullet energy of the tank NPC, the return value variable name is power and the type is double, so this paper designs the reward rules as follows:

$$reward = \begin{cases} -1 & \text{, When shooting;} \\ 4 * power & \text{, if hitting the target;} \\ -6 * power & \text{, if it was hit.} \end{cases} \tag{6}$$

(a) Experiment platform



(b) NPC movement diagram

Fig.2 Figure of the experimental platform

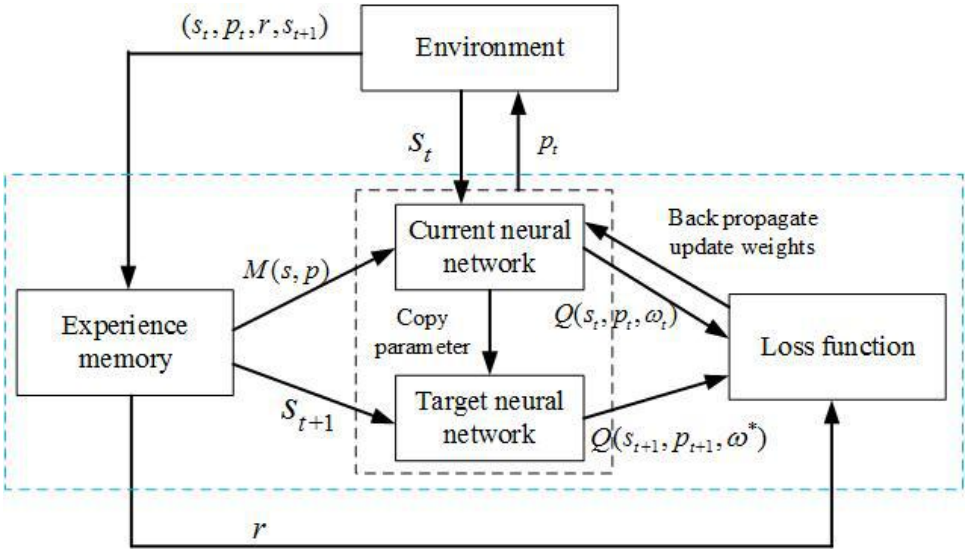


Fig.3 Execution flowchart of Sarsa learning based on double artificial neural network with experience replay

B. *Experimental results and analysis*

We set up the experimental parameters before the experiment, and the update step number of target neural network of DB-SARSA algorithm and RDB-SARSA algorithm τ is 10. The number of updating the current neural network parameters $N=5$, learning rate $\alpha=0.1$, discount rate $\gamma=0.8$, and the threshold of $\varepsilon-greedy$ algorithm $\varepsilon=0.15$. Then we start the experiment, each session process records an experimental result. This paper sets the Average score of the tank NPC hitting the enemy during a session to the total score of hitting the enemy /100. Figure 5 shows the experimental results, in which figure 5(a) shows the number of session in its horizontal axis, and the vertical axis represents the average score of hitting the enemy in each session. We set tank Survival score in each session to the number of survival of the round. The horizontal axis of figure 5(b) represents the number of session, and the vertical axis represents the Survival score of each session of the robotic tank NPC.

It can be seen from the experimental results that the NPC trained by RDB-SARSA algorithm has the highest average score in every session, and the RDB-SARSA algorithm begins to converge after 33 sessions. The DB-SARSA algorithm begins to converge after 42 sessions, and average score of SARSA algorithm is the lowest. The NPC trained by SARSA algorithm after 45 sessions began to converge. This is because the DB-SARSA algorithm using neural network approximates value function, and it can avoid the Shortcomings of the classic Sarsa algorithm to find all states, so which convergence speed is more faster. The RDB-SARSA algorithm which uses the experience replay technology can use the existing experience so that the convergence speed is the fastest and the learning ability is stronger, so the average score is the highest. The curve of survival score of the NPC and the curve of average score have similar characteristics. In the curve of survival score of NPC, the figure5(b), the convergence session of three methods is the same as the curve of average score. The survival score of the NPC trained by the RDB-SARSA algorithm is significantly higher than that of the other two methods, and which have the fastest convergence rate.

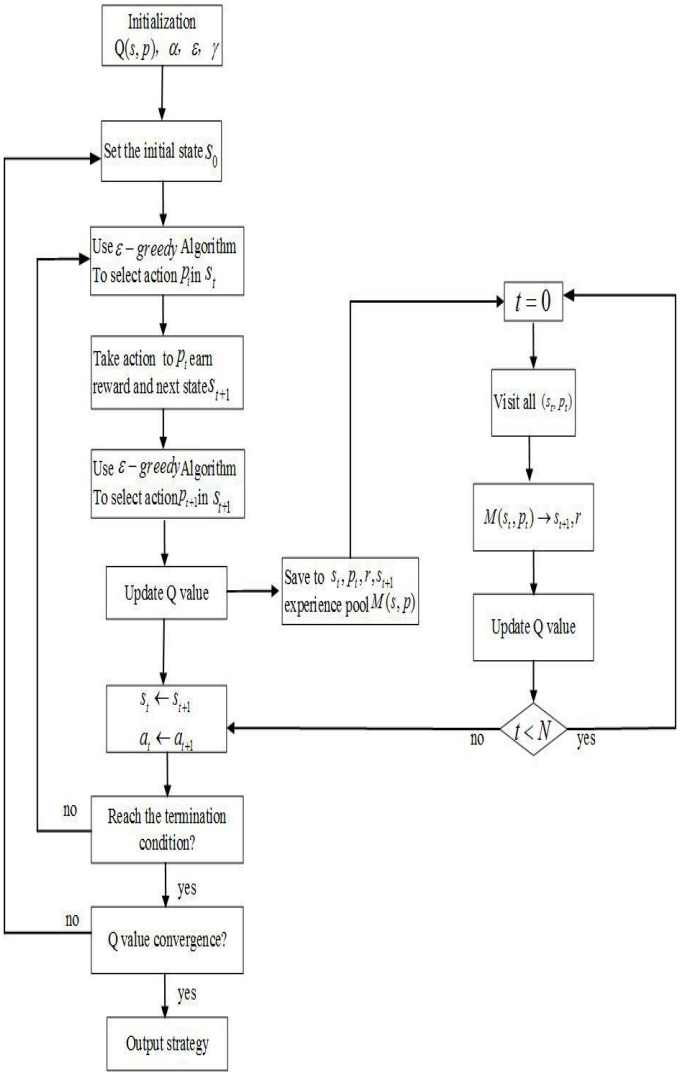
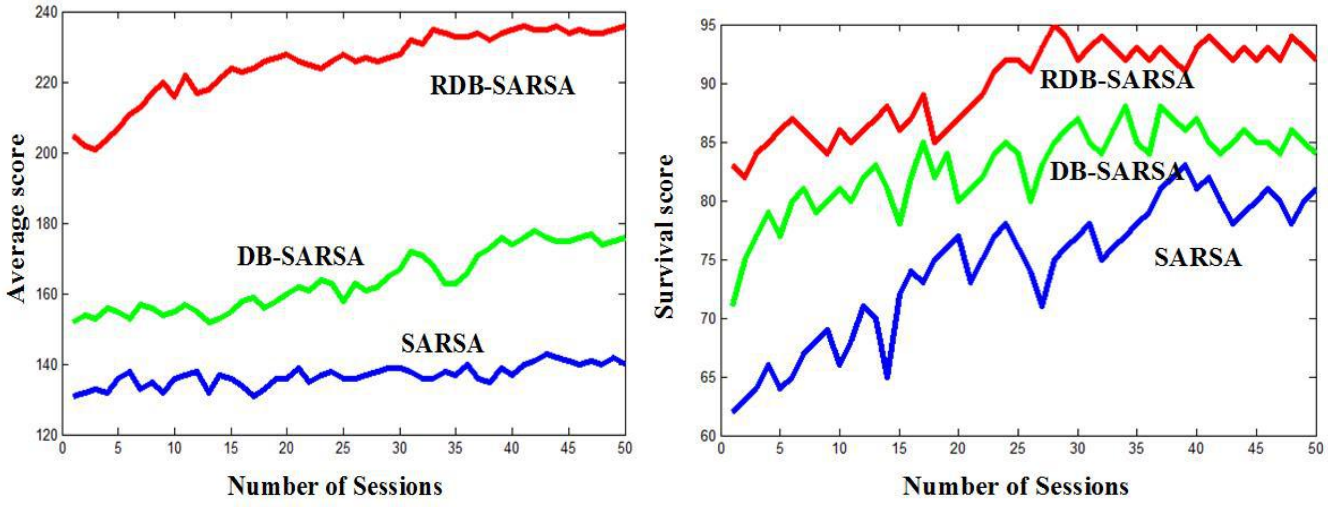


Fig.4 Execution flowchart of Sarsa algorithm with experience replay



(a) The curve of the average score

(b) The curve of survival score

Fig.5 Figure of the experimental results

V. CONCLUSION

In this paper, a Sarsa learning method based on neural networks with experience replay is presented, which is designed in the Robocode platform. It can be seen from the experimental results that the scores of NPC trained by this method are significantly higher than those trained by Sarsa learning based on neural network and those trained by classical Sarsa learning.The NPC trained by the method which is presented in this paper has a greater ability to defend itself and higher survival score .The validity of the proposed method is proved.

REFERENCES

[1] Vinyals O, Ewalds T, Bartunov S, et al. StarCraft II: A New Challenge for Reinforcement Learning[J]. 2017.

[2] Andersen P A, Goodwin M, Granmo O C. Towards a Deep Reinforcement Learning Approach for Tower Line Wars[J]. 2017.

[3] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J]. Computer Science, 2013.

[4] Patel P G, Carver N, Rahimi S. Tuning computer gaming agents using Q-learning[C]// Computer Science and Information Systems. IEEE, 2011:581-588.

[5] Phon-Amnuaisuk S. Learning Chasing Behaviours of Non-Player Characters in Games Using SARSA[M]// Applications of Evolutionary Computation. Springer Berlin Heidelberg, 2011:133-142.

[6] Yasini S, Sitani M B N, Kirampor A. Reinforcement learning and neural networks for multi-agent nonzero-sum games of nonlinear constrained-input systems[J]. International Journal of Machine Learning & Cybernetics, 2016, 7(6):967-980.

[7] Al-Tamimi A, Lewis F L, Abu-Khalaf M. Model-free Q-learning designs for discrete-time zero-sum games with application to H-infinity control[C]// Control Conference. IEEE, 2015:1668-1675.

[8] Corazza M, Sangalli A. Q-Learning and SARSA: A Comparison between Two Intelligent Stochastic Control Approaches for Financial Trading[J]. Working Papers, 2015, 15(2):1-2.

[9] Hartness K. Robocode: using games to teach artificial intelligence[J]. Journal of Computing Sciences in Colleges, 2004, 19(4):287-291.