Reinforcement Learning from Algorithm Model to Industry Innovation: A Foundation Stone of Future Artificial Intelligence

DONG Shaokang, CHEN Jiarui, LIU Yong, BAO Tianyi, and GAO Yang

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210008, China)

DOI: 10.12142/ZTECOM.201903006

http://kns.cnki.net/kcms/detail/34.1294.TN.20190920.2105.006.html, published online September 20, 2019

Manuscript received: 2019-07-10

Abstract: Reinforcement learning (RL) algorithm has been introduced for several decades, which becomes a paradigm in sequential decision-making and control. The development of reinforcement learning, especially in recent years, has enabled this algorithm to be applied in many industry fields, such as robotics, medical intelligence, and games. This paper first introduces the history and background of reinforcement learning, and then illustrates the industrial application and open source platforms. After that, the successful applications from AlphaGo to AlphaZero and future reinforcement learning technique are focused on. Finally, the artificial intelligence for complex interaction (e.g., stochastic environment, multiple players, selfish behavior, and distributes optimization) is considered and this paper concludes with the highlight and outlook of future general artificial intelligence.

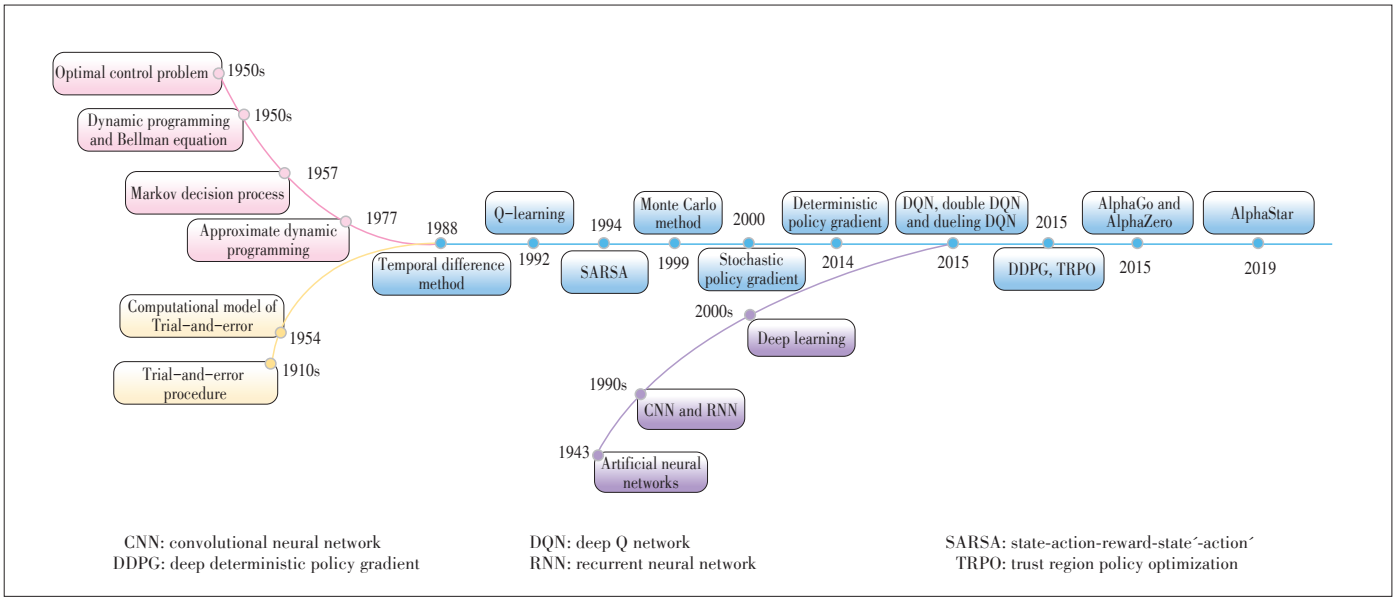
Keywords: artificial intelligence; decision-making and control problems; reinforcement learning

1 Introduction

Reinforcement learning (RL) originates from the trial-and-error (TE) procedure, which was first conducted in the animal learning by Thorndike in 1998 [1]. In 1954, the first concept of the computational model of TE was introduced [2]. Another term "optimal control" was first used to control a dynamical system to reach the goal of a controller. In the 1950s, Bellman introduced the dynamic programming (DP) method to solve the optimal control problems by the Bellman equation [3] and expressed this kind of control problems as Markov decision processes (MDPs) [4]. Later on, Werbos [5] introduced an approximate dynamic programming method as adaptive critic designs (ACDs) in 1977. After a decade, Sutton introduced the temporal difference (TD) method [6] in 1988, marking the point at which the TE procedure and the control problem became interrelated and then produced the field of traditional reinforcement learning. In 1992, Watkins combined the TD learning and MDPs together to solve the optimal policy, which is the well-known algorithm Q-learning [7]. In 1994, Rummery introduced the on-policy version of TD learning as the state-action-reward-state´-action´ (SARSA) algorithm [8]. As for a general condition in MDPs, Thrum introduced a partially observable Markov decision process (POMDPs) and designed the Monte Carlo method [9] to solve it in 1999. At the same time, Sutton introduced another policy-based perspective to solve the reinforcement learning problems and proved the stochastic policy gradient (SPG) [10] computational formula. In 2014, David Silver introduced the deterministic policy gradient (DPG) [11], the essence of which is to maximize the Q value function.

With the rapid development of deep learning, Google DeepMind utilized the deep network to approximate Q value function to address the problem of continuous state space, which was the origin of modern deep reinforcement learning and called deep Q network (DQN) [12]. After that, several improvements related to DQN emerged, such as double Q-network [13], prioritized experience replay [14] and dueling network [15]. In 2015, Google combined DQN and DPG to introduce the deep deterministic policy gradient (DDPG) [16], which extended the deep reinforcement learning (DRL) method to continuous action space control problem. Besides that, Google Gorilla introduced the asynchronous distributed reinforcement learning framework [17] in 2015. Further, John Schulman introduced Trust Region Policy Optimization (TRPO) [18] that is effective for optimizing large nonlinear policies. The development trajectory is shown in **Fig. 1**.

In the industry filed, Google's DeepMind designed the AlphaGo and the AlphaZero to beat the best professional Go players KE Jie and LEE Sadol [19]. Then, more and more Atari games and MuJuCo physic problems [20] and 3D maze games [21] got great scores through deep reinforcement learning. In 2017, OpenAI announced that the reinforcement learning agent could beat the best game player on the online game Dota 2. In 2018, Tencent introduced a hierarchical macro strategy model [22] for a popular 5v5 multiplayer online battle (MOBA)



▲Figure 1. The development trajectory of reinforcement learning technique.

game "Honor of Kings" to achieve a 48% winning rate against human player teams which are ranked top 1%. In 2019, Google's DeepMind designed AlphaStar to beat two professional players in Warcraft II.

2 Background

2.1 Preliminary

Reinforcement learning is a field of machine learning inspired by psychology, in which the optimal control problem focuses on how to make the agent take different actions in an environment to maximize the cumulative long-term returns. In most situations, it is not possible for the agents to know the optimal action directly. In other words, the agent needs to interact with the environment to learn the optimal policy with the immediate reward feedback given and its reliability to the environment makes the study challenging and interesting. The interactions between the agent and environment are described by three essential elements: state , action a and reward r , the relationship of which is illustrated in **Fig. 2**. The state of the environment in the time step t is S_t. At the time the agent takes an action A_t. Then the environment feedbacks a reward R_t and transits to another state S_{t+1}.

The reinforcement learning task as Fig. 2 can be formulated as a MDP $M=<S,A,T,R>$, where S is the state set of environment, A is the action set, $T:S\times A\times S\rightarrow[0,1]$ is the state transition probability and $R:S\times A\rightarrow\mathbb{R}$ is the immediate reward. A policy $\pi:S\times A\rightarrow[0,1]$ is a mapping from states to probabilities of selecting each possible action. If the agent follows a policy π, the probability of taking action a in state s is $\pi(a|s)$. The goal of an agent is to learn the optimal policy π^* in order to get more long-term returns $G=\sum_{t=0}^{\infty}\gamma^t r_t$, where $\gamma\in[0,1]$ is the discounted factor. The agent becomes farsighted when γ approaches 1 while the agent becomes shortsighted when γ is close to 0.

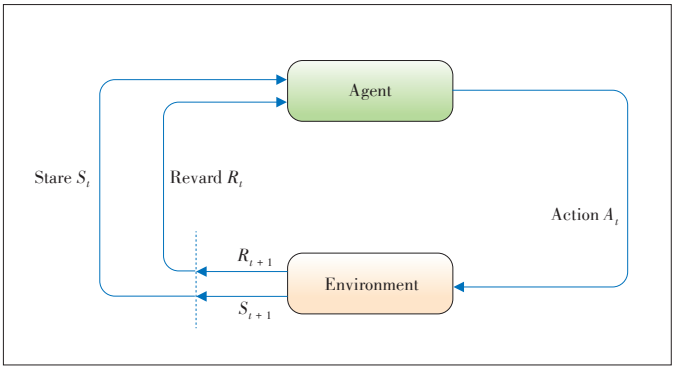
The state value function is defined to measure the utility of a state. For the MDP above, the state value function under the policy π is defined as

$$V_\pi(s)=\mathbb{E}_\pi\left[G_t|S_t=s\right]=\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t|S_t=s\right], \quad (1)$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of the random variable return when the agent is under the policy π. Similarly, the state-action value function is defined as

$$Q_\pi(s,a)=\mathbb{E}_\pi\left[G_t|S_t=s,A_t=a\right]=\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t|S_t=s,A_t=a\right]. \quad (2)$$

The agent always tries to find the optimal policy. A policy is better than or equal to other policies if and only if the state val-



▲Figure 2. The interaction between the agent and environment.

ue function under this policy is greater than or equal to that under other policies. In other words, the mathematical expression can be formulated as

$$\pi \geqslant \pi' \Leftrightarrow V_{\pi}(s) \geqslant V_{\pi'}(s), \forall s. \tag{3}$$

Therefore, there always exists at least one policy that is better than or equal to all the others, which is defined as the optimal policy π^*. The corresponding state value function and state-action value function is denoted as

$$V^*(s) = \max_{\pi} V_{\pi}(s) \forall s, \tag{4}$$

$$Q^*(s,a) = \max_{\pi} Q_{\pi}(s,a) \forall (s,a). \tag{5}$$

Therefore, all kinds of reinforcement learning algorithms tend to obtain the optimal policy. Based on the prior knowledge of reward R and dynamic T, we can classify the methods into model-based and model-free. The model-based methods can have access to all the information of MDP, in which the well-known algorithm is dynamic programming (illustrated in the next section). The model-free methods contain the Monte Carlo and the temporal difference algorithms. The temporal difference algorithms can further be classified as the value-based and the policy-based. The value-based methods find the optimal policy through value function, in which the famous algorithm is SARSA and Q-learning depending on whether it is on-policy or off-policy. The other kind of policy-based algorithms converges to the optimal policy through policy gradient. Besides that, the policy-based and value-based methods can be combined into the actor-critic methods. Through the deep learning network, Q-learning and actor-critic methods can be extended to DQN and DDPG in order to address large-scale problems. The classification of different reinforcement learning methods is shown in **Fig. 3**.

2.2 Reinforcement Learning Methods

In this section, we will illustrate the three basic reinforcement learning methods: dynamic programming, Monte Carlo, and temporal difference. The dynamic programming method needs information about the reward and dynamic of the environment, while the other two are model-free methods.

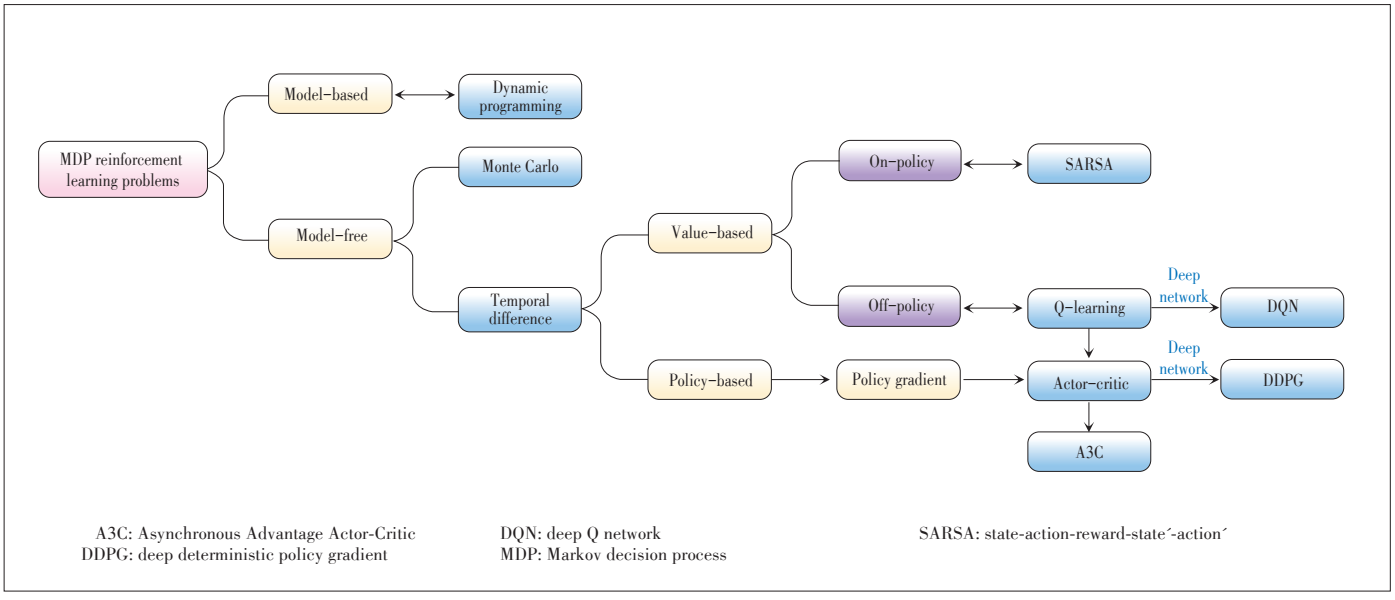
2.2.1 Dynamic Programming Method

The key idea of DP is utilizing the Bellman equation. We can utilize DP to compute the value function illustrated above. Based on the definition of the state value function, we can obtain the Bellman optimality equation as

$$\begin{aligned} V^*(s) &= \max_{a \in A(s)} Q_{\pi^*}(s,a) = \\ &\max_{a \in A(s)} \mathbb{E}_{\pi^*}\left[G_t | S_t = s, A_t = a \right] = \\ &\max_{a \in A(s)} \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a \right] = \\ &\max_{a \in A(s)} \mathbb{E}_{\pi}\left[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a \right] = \\ &\max_{a \in A(s)} \sum_{s',r} p(s',r|s,a) \left[r + \gamma V^*(s') \right], \end{aligned} \tag{6}$$

where $p(s',r|s,a)$ is the state transition probability under the reward r, and $\gamma \in [0,1]$ is the discounted factor. Obviously, we can obtain that the state transition probability $p(s'|s,a) = \sum_{r \in R} p(s',r|s,a)$. Then we will illustrate how to use DP to converge to the optimal policy.

In the first step, we should compute the state value function V_{π} for any arbitrary policy π, which is called the policy evaluation. Similar to Eq. (6), we can obtain the iterative equation as



▲**Figure 3. The classification of different reinforcement learning methods.**

the updated rule

$$V_{k+1}(s)=\mathbb{E}_{\pi}\left[R_{t+1}+\gamma V_k(S_{t+1})|S_t=s\right]=$$
$$\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r+\gamma V_k(s')\right]. \quad (7)$$

In this case, we can rigorously prove that $V_k=V_{\pi}$ is a fixed point for this iterative equation. Therefore, the updated equation can converge to the state value function under any arbitrary policy π.

Then, we need to compute the optimal policy instead of an arbitrary policy, called the policy improvement. Similarly, we can take the action as a greedy way to obtain a new policy π' that is better than or equal to the current policy π as

$$\pi'(s)=\arg\max_a Q_{\pi}(s,a)=$$
$$\arg\max_a \mathbb{E}_{\pi}\left[R_{t+1}+\gamma V_{\pi}(S_{t+1})|S_t=s,A_t=a\right]=$$
$$\arg\max_a \sum_{s',r} p(s',r|s,a)\left[r+\gamma V_{\pi}(s')\right]. \quad (8)$$

It can be proved that the greedy policy π' is better than the current policy π as $V_{\pi}(s)\leqslant V_{\pi'}(s),\forall s$. So far, we can utilize the policy evaluation to get the stable value function and then policy improvement to obtain a better policy until the state value function and policy are both stable, which is called the policy iteration. Then, the current policy is obviously the optimal policy and the value function is the optimal value function.

2.2.2 Monte Carlo Method

Monte Carlo method is a kind of model-free method, so we cannot utilize the reward and state transition probability to compute the state value function. Instead, the state value function is estimated by repeatedly generating episodes and averaging the returns under policy π as

$$V_{\pi}^{MC}(s)=\mathbb{E}_{\pi}\left[G_t|S_t=s\right]. \quad (9)$$

Although we do not utilize the information related to the model, it must satisfy the assumption that for any states and actions, the sampling times must approach infinity. Only under that condition, we can ensure that the Monte Carlo value function V_{π}^{MC} converges to the true state value function. Then, we need to modify the greedy method as ϵ -greedy to take the policy improvement in order to guarantee the assumption:

$$\pi'(a|s)=\begin{cases}1-\epsilon+\epsilon/|A(S)|, if\, a=\arg\max_a Q_{\pi}(s,a_k)\\ \epsilon/|A(s)|, if\, a\neq\arg\max_k Q_{\pi}(s,a_k)\end{cases}, \quad (10)$$

where $|A(s)|$ denotes the number of actions taken in the state s. In this case, we can trade off the exploration-exploitation dilemma, but the policy of evaluation and exploration is different, called the off-policy. The fact is that the off-policy method is sometimes unstable in continuous state-space problems. Therefore, we usually design the on-policy method through importance sampling.

2.2.3 Temporal Difference Value-Based Method

Similar to the Monte Carlo method, the temporal difference method is also a model-free method. However, what makes it different from the Monte Carlo method is that it needs bootstrapping, which means the estimated value function V_{k+1} is updated based on the last estimated value function V_k. The simplest temporal difference update is that

$$V(s_t)\leftarrow V(s_t)+\alpha\left[R_{t+1}+\gamma V(s_{t+1})-V(s_t)\right], \quad (11)$$

where $\alpha\in(0,1)$ is the learning rate. As the category in Monte Carlo, the temporal difference method is also divided into on-policy and off-policy, which are the well-known algorithms SARSA and Q-learning.

In SARSA, we update the state-action value function for the current policy and take the action based on the state-action value function. The algorithm can converge to an optimal policy with probability 1 as long as all states and actions are visited an infinite number of times. The update equation is

$$Q(S,A)\leftarrow Q(S,A)+\alpha\left[R+\gamma Q(S',A')-Q(S,A)\right], \quad (12)$$

where S' and A' denote the next state and next action. It is obvious that the policy of evaluation and exploration is the same, so SARSA belongs to the on-policy methods.

Q-learning is an off-policy method because the evaluation of state-action value function is maximizing over all those actions possible in the next state instead of the current Q function. The update equation is

$$Q(S,A)\leftarrow Q(S,A)+\alpha\left[R+\gamma\max_a Q(S',a)-Q(S,A)\right], \quad (13)$$

where S' denotes the next state and next action. The convergence condition of Q-learning is the same as SARSA.

2.2.4 Temporal Difference Policy-Based Method

Although the value-based methods have made great achievement in many fields, such as robotics and video games, they cannot be applied in the situation of continuous state and action space. In this situation, there exists a new policy-based method, in which the policy is explicitly represented by the function approximator and updated based on the gradient of expected reward [10]. Let $J(\pi_0)$ denote the performance objective as an expectation as

$$J(\pi_0)=\int_S \rho^{\pi}(s)\int_A \pi_0(a|s)\,r(s,a)\,da\,ds=$$
$$\mathbb{E}_{s\sim\rho^{\pi},a\sim\pi\theta}\left[r(s,a)\right], \quad (14)$$

where $\rho^{\pi}(s)$ is the discounted state distribution as $\rho^{\pi}(s')=\int_s\sum_{t=1}^{\infty}\gamma^{t-1}p_1(s)\,p(s\rightarrow s',t,\pi)\,ds$. Policy gradient algorithms search for a local maximum in $J(\pi_0)$ by ascending the gradient of the policy w.r.t. parameter θ as $\triangle\theta=\alpha\nabla_{\theta}J(\pi_{\theta})$.

Sutton has proved the update rule of stochastic policy gradient as

$$\nabla_\theta J(\pi_\theta)=\int_S \rho^\pi(s)\int_A \nabla_\theta \pi_\theta(a|s)\,Q^\pi(s,a)\,da\,ds=\mathbb{E}_{s\sim\rho^\pi,a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\,Q^\pi(s,a)\right]. \quad (15)$$

The stochastic policy gradient is surprisingly simple and does not depend on the gradient of the state distribution [10]. In 2014, David Silver [11] introduced DPG as

$$\nabla_\theta J(\pi_\theta)=\int_S \rho^\pi(s)\nabla_\theta \pi_\theta(s)\nabla_a Q^\pi(s,a)|_{a=\pi\theta(s)}\,ds=\mathbb{E}_{S\sim\rho^\pi}\left[\nabla_\theta \pi_\theta(s)\nabla_a Q^\pi(s,a)|_{a=\pi\theta(s)}\right]. \quad (16)$$

The difference of DPG and SPG is that the essence of deterministic policy gradient is to maximize the Q value function, which is similar to the value-based method.

2.2.5 Temporal Difference Actor-Critic Method

Actor-critic method combines the value-based method and policy-based method, in which the actor network (policy network) outputs the action selection and critic network (value network) evaluates the performance of action. The well-known DDPG algorithm utilizes the actor-critic idea to address the continuous control problems [16]. In the part of actor, the parameter θ^Q is updated to maximize Q value and parameter θ^μ is updated to make actor more likely to select this action. Therefore, the updated rule is as

$$\nabla_{\theta_\mu} J\approx\frac{1}{N}\sum_i \nabla_a Q^\pi(s,a|\theta^Q)\Big|_{s=s_i,a=\mu(s_i)}\nabla_{\theta_\mu}\mu(s,\theta^\mu)\Big|_{s_i}. \quad (17)$$

As for the part of critic, it is similar to the DQN (illustrated in the next section). The parameter is updated to minimize the TD loss

$$L=\frac{1}{N}\sum_i (y_i-Q(s_i,a_i|\theta^Q))^2, \quad (18)$$

where $y_i=r_i+\gamma Q'\left(s_{i+1},\mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$.

Another benchmarking work Asynchronous Advantage Actor-Critic (A3C) [17] is desired to solve the convergence problem of actor-critic method. It creates multiple parallel environments, in which there exist many thread-specific agents updating the global parameters together. The agents in parallel do not interfere with each other, and the parameter update of the main structure is disturbed by the discontinuity of the update submitted by the thread-specific agents, so the correlation of the update is reduced to accelerate the convergence rate. Different to DDPG, A3C updates the policy gradient through advantage as

$$\nabla_\theta J=\nabla_\theta \log \pi(a_i|s_i;\theta)\,A(s_t,a_t;\theta_v), \quad (19)$$

where the advantage is computed by N-steps TD error as $A(s_t,a_t;\theta_v)=\sum_{i=0}^{N-1}\gamma^i r_{t+i}+\gamma^N V(s_{t+k};\theta_v)-V(s_t;\theta_v)$ and the parameter θ_v represents for each local thread-specific agent.

Because the learning rate of policy gradient is dependent on the specific environment and not easily determined, PPO [23], [24] utilizes the ratio of new policy and old policy in order to restrict the update. The critic network is also similar to the AC3 method to minimize TD error, while the actor network updates the parameter by KL penalty as

$$\nabla_\theta J=\sum_{t=1}^N \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\hat{A}_t-\lambda KL[\pi_{old}|\pi_\theta], \quad (20)$$

where $\hat{A}_t$ is also the estimated advantages as A3C.

After illustrating these reinforcement learning algorithms, we summarize the characteristics of the algorithms and differentiate them based on the policy, bootstrapping and model property in **Table. 1**.

2.3 Deep Reinforcement Learning

Deep learning can help reinforcement learning address the high-dimension problem when the states and actions are continuous variables in the environment. In 2015, Mnih [12] trains a deep neural network to develop a novel artificial agent to interact with the environment, as shown in **Fig. 4**. In other words, DQN utilizes the neural networks to interpret the complex state representation instead of the traditional MDP. Of course, the goal of network to optimize also satisfies the Bellman equation, of which the loss function is defined as

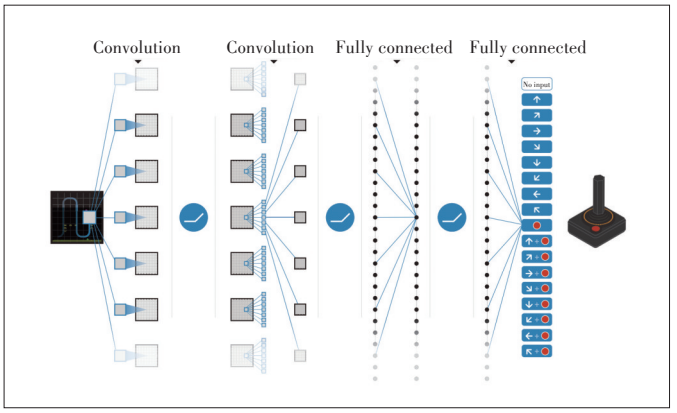
$$L(\theta)=\left(r+\gamma\max_{a'}\hat{Q}(s',a';\theta^-)-Q(s,a;\theta)\right)^2, \quad (21)$$

where $\hat{Q}$ is the target value function and Q is the evaluation val-

▼**Table 1. Comparisons between reinforcement learning (RL) methods**

Category	DP	On-policy MC	Off-policy MC	SARSA	Q-learning
Model-based	√				
Model-free		√	√	√	√
On-policy		√		√	
Off-policy			√		√
Bootstrapping	√			√	√

DP: dynamic programming MC: Monte Carlo
SARSA: state-action-reward-state´-action´



▲**Figure 4. Neural network of the deep Q-learning [12].**

ue function. The purpose of designing the two value function is to eliminate the correlation between the samples due to the unstable property of neural networks. Besides that, generated samples from the environment are stored in the experience replay memory, the data of which are randomly fed to train the network. Note that, only the evaluation network with the parameter β is trained, the target network with the parameter β' is just a duplicate of β but with the delayed update as $\beta' \leftarrow \beta$ for every N steps.

With the development of different scenario requirements, DQN also has some variants. The first algorithm is called double deep Q-network (DDQN) [13], the idea of which is to separate the optimal action with the evaluation step in order to avoid the overestimation of Q-value function. Therefore, the loss function is modified as

$$L(\theta)=\left(r+\gamma\hat{Q}\left(s',\max_{a'}\hat{Q}(s',a';\theta^{-});\theta^{-}\right)-Q(s,a;\theta)\right)^{2}. \quad (22)$$

Besides that, Tom Schaul [14] proposes a prioritized experience replay method to reinforce the goal-related samples with greater TD error. The corresponding sampling probability is the normalization of the absolute TD error as $p_i=|\delta_i|=|r+\gamma\max_{a'}\hat{Q}(s',a',\theta^{-})-Q(s,a;\theta)|$. In addition, Ziyu Wang [15] introduces a dueling network to represent the state value function and state-dependent action advantage function. Then the Q-value function is computed as

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+\left(A(s,a;\theta,\alpha)-\frac{1}{|A|}\sum_{a}A(s,a';\theta,\alpha)\right), \quad (23)$$

where θ denotes the parameter of the convolutional layers, while α and β are the parameters of the two streams of fully-connected layers.

Therefore, it is obvious that the development of reinforcement learning is tracking the increasing demands when applied to the real industry field. In the following sections, we will illustrate the application and open source platforms about reinforcement learning.

3 Reinforcement Learning Industry and Application

3.1 Famous Pioneering Companies

On top of all the research companies, DeepMind is the first that should be mentioned, which was founded in London in 2010 and had been acquired by Google in 2014. The core research filed is about health artificial intelligence (AI), the energy efficiency of Google's data centers and ethics & society. The significant milestones for reinforcement learning research are introducing the AlphaGo, AlphaZero, AlphaStar and Reinforcement Learning with TensorFlow (TFRL) library. The library collects key algorithmic components that we have used internally for a large number of our most successful agents such as DQN, DDPG, and the Importance of Weighted Actor Learner Architecture. Another company OSARO was founded in 2015 in San Francisco, which is devoted to developing products based on proprietary deep reinforcement learning technology in order to process large amounts of unstructured data and efficiently learn complex control tasks. The company VocalIQ was formed in March 2011 and had been acquired by Apple to exploit technology developed by the Spoken Dialogue Systems Group at University of Cambridge. VocalIQ builds a platform for voice interfaces, making it easy for everybody to voice-enable their devices and apps. AI is extremely difficult to be applied in business because the systems cannot start learning on their own. The company CogitAI is solving this problem by self-learning AI platform that learns through its interaction with the real world, which will make the perception and behavior of daily routine become more intelligent and experienced. The Japanese company Preferred Networks was founded in 2014 and then invested by Toyota, with the main products related to automatic driving, medical health and AI in the manufacturing industry. At last, the World's First Deep Reinforcement Learning Platform for the Enterprise Bonsai was acquired by Microsoft in 2018, which focuses on optimizing the Azure public cloud service platform.

3.2 Representative Applications

Nowadays, the reinforcement learning techniques have been applied in many industries including robotics, automatic driving, natural language processing, computer vision, finance, healthcare, smart grid, intelligent transportation systems, and so on. In robotics, deep reinforcement learning can help to address the high-dimensional control problems, which has become not only a major research topic but also an efficient method to build products for industrial robots. Usually, the RL can train to control robotic arms and legs to accomplish the carrying or motion tasks. A famous experiment about the autonomous helicopter is designed by imitation learning, a branch of reinforcement learning. With a higher level goal, reinforcement learning can also be applied in automatic driving. Only relying on clever models with high-quality training data and carefully thinking out objective functions, the agent can learn a more comprehensive set of skills for driving. Up to now, the Tesla automatic driving cars can independently achieve the cruise, steering, line-changing and parking tasks, which have been produced and put into application in some areas.

As in natural language processing, the researchers face a sequential prediction problem where future observations (visited states) depend on previous actions. This is challenging because it violates the common independently and identically distributed (i.i.d.) assumptions made in statistical learning. For example, naively training the agent on the gold labels alone would unrealistically teach the agent to make decisions under the assumption that all previous decisions were correct, potentially causing it to over-rely on information from past actions [25].

Especially for the multiple rounds of question & answer and semantic analysis system, reinforcement learning methods can achieve great performance because the context semantic information can be considered. Therefore, it is clearly seen that the RL model is giving more human-like answers than tradition systems. Yangfeng Ji [26] proposes an adversarial learning procedure to where they train a generative model to produce response sequences and a discriminator to distinguish between the human-generated dialogues and the machine-generated ones. The outputs from the discriminator are then used as rewards for the generator, pushing the system to generate human-like dialogues.

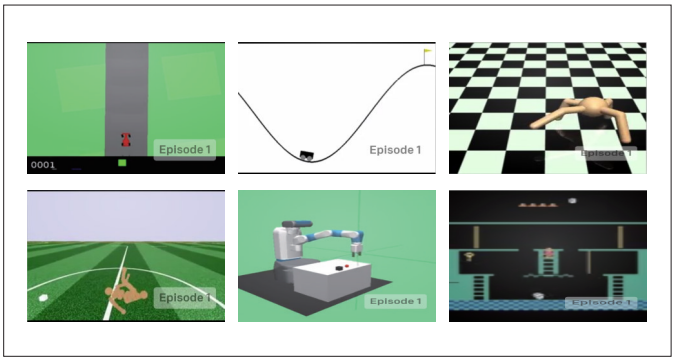
There is a huge potential for reinforcement learning in finance. Apart from just playing games, it seems reasonable for such a framework to have meaningful applications in finance and trading due to the following reasons: the size of states in the finance environment may be large and continuous. Actions may have long-term consequences, which means the agent should have further insight. Your trader actions can affect the current market environments. Financial management is the continuous investment of new funds into different financial products with the goal of achieving maximum returns. Therefore, when facing these challenges, the reinforcement learning framework is an effective tool to provide smarter solutions for fund management. Another application is in the search and recommender systems, which usually utilize the multi-arm bandit (MAB, a special reinforcement learning method) algorithm to address the exploration-exploitation dilemma. Then Alibaba models the browsing and purchasing behavior of users as MDP in the search scene [27]. What takes Taobao search to the next level is the 20% improvement of the performance in the real-time learning and decision computing system using reinforcement learning. In the recommended scenario, Alibaba applies deep reinforcement learning and adaptive online learning to analyze the users' behaviors and the commodity characteristics, which helps users find devoted goods quickly and improves the effect index by 10%–20%.

Finally, a closely related application to everyday life is healthcare. Nearly all large companies in the healthcare field have already begun to use deep reinforcement learning technology in practice. Researchers have also found the potential of reinforcement learning in the medical image screening for diagnosis detection, medical chatbots, and clinical decision-making simulation. For example, in robot-assisted surgery, doctors usually guide the robot to operate instruments by remote control. With the applications of computer vision models to observe the surgical environment and reinforcement learning methods to learn the surgeon´s movements, the robustness and adaptability of robot-assisted surgery are effectively improved. Additionally, reinforcement learning can also enhance the efficiency of ICU rescue. When rescuing and caring for intensive care patients, doctors are often caught in a dilemma: blood test indicators can provide critical information for patient rescue, but too frequent tests may aggravate the risk of illness and increase the cost of treatment. A research group from Princeton University finds that the reinforcement learning algorithm mentioned above can improve the ability of machine learning to select clinical test items and get the most rewards by optimizing the order of clinical test. Therefore, reinforcement learning is a practical tool that can help companies optimize their service provision, improve the standard of care, generate more revenue, and decrease risk.

4 Open Source Platform

Reinforcement learning has become a hot topic all over the world, but where can we learn and practice the RL algorithms? The good news is that many companies have opened up resources and platforms. The famous OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms, which includes a series of continuously growing and improving environments (e.g. simulated robots and Atari). It helps teach agents to do everything, such as walking and playing Ping Pong. In addition, Gym is compatible with other numerical calculation libraries, such as tensorflow and theano library. The environments include different scenes, such as Atari, Box2D, Classic control, MuJoCo, Roboschool, Robotics, and Toy text. **Fig. 5** shows several representative games in Gym.

Arcade Learning Environment (ALE) is an object-oriented open source platform, which provides an interface to hundreds of Atari 2600 game environments, each one different, interesting, and designed to be a challenge for human players [28]. ALE is easy to add any game agent based on the object-oriented framework. In addition, the simulation core is decoupled from the game screen rendering and sound module to improve the simulation efficiency. It supports multiple operating systems (OS X, Linux) and can be cross-language developed. Finally, the platform also has some rules to evaluate different algorithms. Another platform VizDoom is an artificial intelligence research platform based on the 3D shooting game Doom, which is used for reinforcement learning from original visual



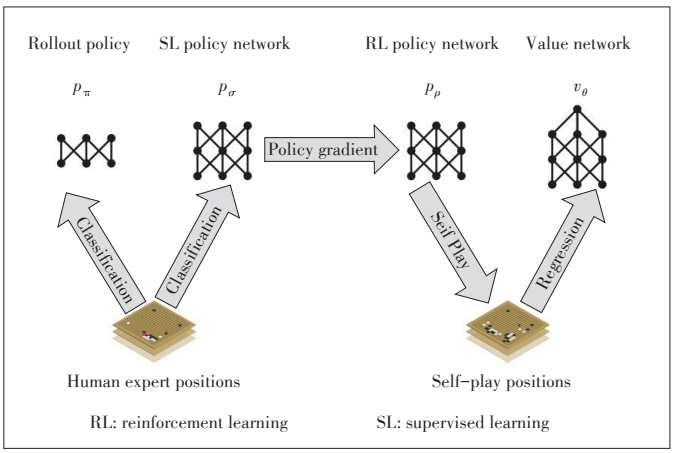
▲ **Figure 5. The Gym games (from left to right and top to bottom: CarRacing, Mountainair, Ant, RoboschoolHumanoidFlagrunHarder, FetchPickAndPlace and MontezumaRevenge).**

information and mainly used to study machine vision learning, especially deep reinforcement learning. And the Google DeepMind Lab is also a first-person perspective 3D gaming platform designed specifically for research into general-purpose artificial intelligence and machine learning systems. It can be used to study how agents learn to perform complex tasks in a large, partially visual and visually diverse environment. The other platform The Open Racing Car Simulator (TORCS) is an open-source 3D car racing simulator available for many systems. TORCS is designed to enable pre-programmed AI drivers to race against one another while allowing the user to control a vehicle using either a keyboard, mouse or wheel input.

The other game platform StarCraft II Learning Environment (SC2LE) is a StarCraft II artificial intelligence research environment jointly released by DeepMind and Blizzard Entertainment, aiming to further promote the development of AI by developing learning systems capable of solving complex problems. SC2LE can also choose the mini-game maps, which can first evaluate the algorithm in a simpler environment. The configurations can set player and time limits, whether to use the game outcome or curriculum score and a handful of other things. It allows Python codes to communicate with the API and this client will be the main focus of this guide, from which we can get spatial features from the map and call actions that mimic how humans interact with the game.

5 An Insight: From AlphaGo to AlphaZero

Several years ago, the milestone in the field of artificial intelligence was the appearance of AlphaGo [19] that beat the professional player Lee Sedol. The challenge of Go for classic methods is its enormous search space and the difficulty in evaluating board positions and moves. The Go game contains approximately $b^d(b \approx 250, d \approx 150)$ possible move sequences, which results in the failure of brute search. Therefore, the AlphaGo combines the supervised learning and reinforcement learning with Monte Carlo tree search (MCTS) to reduce the space. There are four networks in total: rollout policy network, supervised learning (SL) policy network, RL policy network, and value network. The relationship between these networks is shown in **Fig. 6**, in which the rollout and SL policy network are trained by the expert data. The target of them is to predict the most likely moves of human by learning the conditional probability $p(a|s)$, where s denotes the state of the chessboard and a denotes the move position. The difference of them is that the SL policy network is then used to train the RL policy network, while the rollout policy network sacrifices the accuracy for quick simulation in order to provide the possibility of MCTS. The training of SL policy network is based on CNN, where the inputs are the chessboard features (19×19×48) and outputs are the probability of different positions. The training process cost about 3 weeks by 50 Graphics Processing Units (GPUs) through the expert game data in Kiseido Go Server (KGS) platform.



▲**Figure 6. The different networks of AlphaGo [19].**

Another important network is the RL policy network, the basic idea of which is using self-play to formulate the new experience that human cannot image. The detailed steps are as follows:

1) Set the structure of RL policy network identical to the SL policy network and its weights are initialized to the same values.

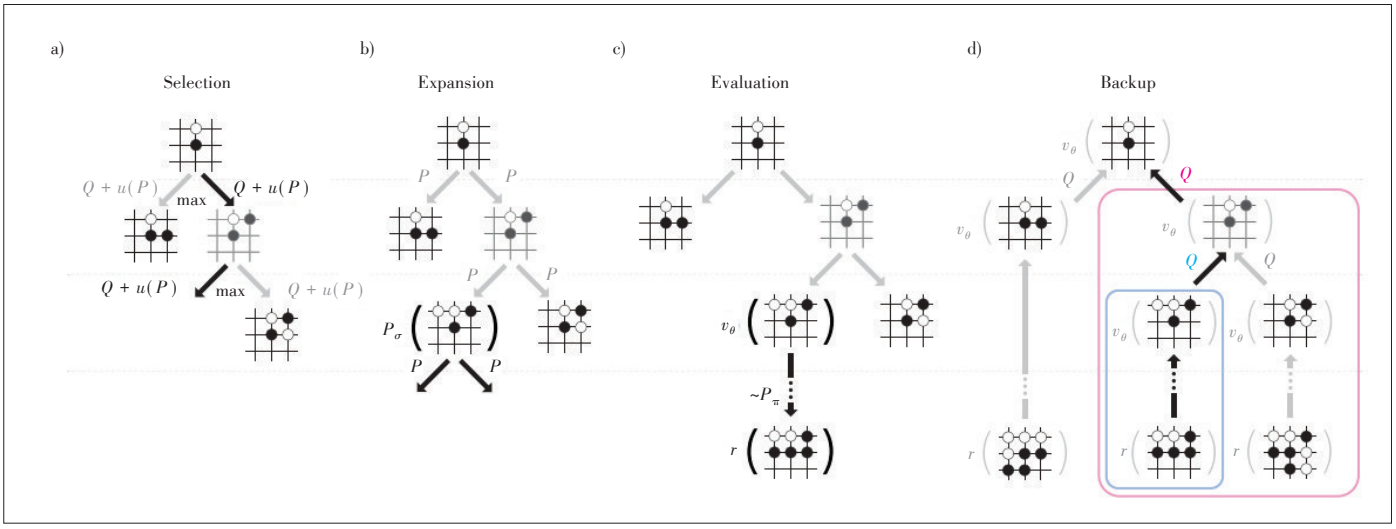
2) Put this network to the opponent pool.

3)Make the current RL policy network to play against a random version in the opponent pool and then utilize the reinforcement algorithm to maximize the expected outcome by updating the parameters as $\Delta\rho \propto \partial \log P_\rho(a_t|s_t) z_t / \partial\rho$, where ρ denotes the parameters in the RL policy network and $z_t = \pm r(s_T)$ is the terminal reward at the end of the game as +1 for winner else −1 for the loser.

4) Put the current network into the opponent pool after every 500 iterations and return to step 2 recursively.

Through the RL training process, the winning rate can achieve 80% against the SL policy network.

The last value network is used to compute the optimal value functions and predict the winner of the games. This neural network has a similar architecture to the policy network and outputs a certain action. The value function is regressed by the parameter θ, which minimizes the mean squared error (MSE) between the predicted values. Finally, the AlphaGo takes actions by the MCTS algorithm (**Fig. 7**). When MCTS is faced with the current state, AlphaGo will simulate some kinds of strategy to play a few steps or all the way to the end by itself. More and more simulation can make the AlphaGo deduce more accurately. Because of the policy network, the search space has been narrowed down to the high-probability actions. The edge of tree stores the basic information such as value function $Q(s,a)$, visited times $N(s,a)$, and probability $P(s,a)$. The first step of MCTS is to select the appropriate actions as $a_t = \arg\max_a(Q(s_t,a) + u(s_t,a))$. What is different from the tradi-



▲Figure 7. The process of MCTS [19].

tional RL methods is the additive term $u(s_t, a) \propto P(s_t, a)/1 + N(s_t, a)$ that is proportional to the prior probability $P(s_t, a)$ and inversely proportional to the visited times $1 + N(s_t, a)$. When the leaf node in L depth is visited at time step L, the expansion step should be conducted and the new leaf node keeps the same probability of its parent node. The third step evaluation is to compute the reward of the leaf node; therefore, we are able to use the value network to compute as well as the rollout network to run to the end and obtain a winning or losing the reward. Finally, we can back up the Q value through the sub-tree as

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i), \tag{24}$$

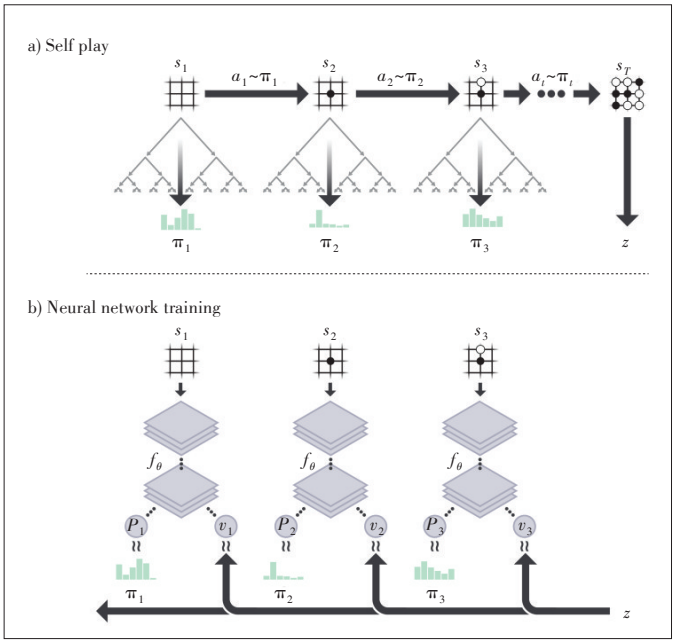
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i), \tag{25}$$

where s_L^i is the leaf node in the i -th simulation and $1(s,a,i)$ indicates whether an edge (s,a) was traversed. Through the backup process, we can recursively take the appropriate action in the selection step.

In conclusion, AlphaGo needs the human experience to predict the position in SL policy network. Combined with the RL policy network, AlphaGo can generate new experience through self-play, which can achieve a better win rate. Besides that, the MCTS and value work can be used to get a more accurate result through the search process. However, the drawback of AlphaGo is that it needs the supervision of human game data, which is expensive or even unavailable. Moreover, human data may also cause an inaccurate bias of the deep network. Another shortcoming is that AlphaGo uses four networks, which is very complex and costs more times to converge. Therefore, AlphaZero [28] is introduced to address these problems.

AlphaZero is learned from a random game, using self-play reinforcing learning, without human data or supervision. The policy and value networks are combined with only a single network, the input of which is the direct position of black and white instead of the manually designed features. The detailed process is shown in **Fig. 8**. A self-play game has many different states $S_1, S_2, S_3, ..., S_T$. In each state, the neural network is training to output the action probability vector $\boldsymbol{P}_t$ and the predicted value v_t. Its goal is to minimize the difference between policy vector $\boldsymbol{P}_t$ and the MCTS probabilities π_t and also that of the predicted value v_t and the end reward z. The loss function is

$$L(\theta) = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{P} + \mathrm{c}\|\theta\|^2. \tag{26}$$



▲Figure 8. The network of AlphaZero [28].

This neural network improves the efficiency of Monte Carlo tree search, which enhances the accuracy of move selection and stronger self-play in the next iteration. The final performance can achieve 100–0 against AlphaGo.

6 Future Techniques

Reinforcement learning has become more and more important, but there are also several problems which need to be addressed in the future. As we all know, deep reinforcement learning has continuous large state and action spaces, resulting in the long endless iteration times to learn a stable converged policy. Moreover, the real-world environment is more complex and nonstationary, because there are many other agents that also need to learn an optimal policy in the environment. Therefore, future reinforcement learning research must contain transfer learning and multi-agent reinforcement learning.

6.1 Transfer in Reinforcement Learning

At present, in many sub-fields of machine learning (such as neural networks and reinforcement learning), transfer learning has made some remarkable progress. In real-world environments, many learning tasks may share some similar features, resulting in similar policies. Therefore, the agent is not necessary to learn from scratch but with a prior bias. To enhance the performance, the research of transfer in reinforcement learning should answer the following questions: whether to transfer, what to transfer, how to transfer and how to evaluate the transfer performance. Whether to transfer depends on the similarity of source and target tasks, which contains the state space, action space, dynamics, and rewards. The transfer methods are usually classified into behavior transfer and knowledge transfer. The behavior transfer means to utilize the previously learned policy and common sub-tasks, including the policy transfer [29], option transfer [30], and hierarchical reinforcement learning [31]. While the knowledge transfer contains the value function transfer [32], relational reinforcement learning [33], and heuristic information transfer [34]. Different from behavior transfer, the knowledge transfer focuses on the essence of tasks and tries to learn a general model to address the problem, which is more similar to our minds but also more difficult. Finally, the common emulation of transfer in reinforcement learning is the jump-start (the initial enhancement), learning speed, and asymptotic performance.

6.2 Multi-Agent Reinforcement Learning and Game

Multi-agent system (MAS) also attracts many researchers, because many complex tasks cannot be modeled as the single-agent environment. For example, the famous prisoner´s dilemma is based on game theory, and complex environments such as Warcraft II are the fields of multi-agent reinforcement learning. In MAS, agents communicate and interact with other agents, of which the MDP can be generalized to a stochastic game, or a Markov game. As usual, the multi-agent reinforcement learning algorithms can be classified into cooperative, competitive and game types, which results in the famous algorithms combining the reinforcement learning with game theory, including minimax-Q [35], Correlated Q-Learning [36], Nash Q-Learning [37], Value-Decomposition Networks (VDN) [38], Mixing Q-network(QMIX) [39], Multi-agent Deep Deterministic Policy Gradient (MADDPG) [40], and so on. However, VDN and QMIX can only work in fully cooperative multi-agent environments. Although MADDPG has become a benchmarking work in multi-agent environments, there are also many challenges in real-world environments. For example, MADDPG cannot deal with the game environment and the opponent model prediction is impractical in partial observability environment. Up to now, the main challenges of multi-agent reinforcement learning are about the non-stationary environment, partial observability MDP (POMDP), training schemes, continuous space, and an equilibrium solution.

7 Conclusions

This paper presents an overview of the background and development of reinforcement learning, industry applications, and open source platforms. Since the remarkable milestone in AI beating professional Go players, we illustrate the famous AlphaGo and AlphaZero in details. Finally, we introduce the future technologies such as transfer reinforcement learning, multi-agent reinforcement learning, and game in complex interaction environment including stochastic environment, multiple players, selfish behavior and distributes optimization. In conclusion, we can look forward that future general artificial intelligence can be achieved through deep learning and reinforcement learning.

References

[1] THORNDIKE E L. Animal Intelligence: An Experimental Study of the Associate Processes in Animals [J]. American Psychologist, 1998, 53(10): 1125. DOI: 10.1037/0003-066X.53.10.1125
[2] MINSKY M L. Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain Model Problem [M]. New Jersey, America: Princeton University Press, 1954
[3] BELLMAN R. On the Theory of Dynamic Programming [J]. Proceedings of the National Academy of Sciences of the United States of America, 1952, 38(8): 716. DOI: 10.1073/pnas.38.8.716
[4] BELLMAN R. A Markovian Decision Process [J]. Journal of Mathematics and Mechanics, 1957: 679–684. DOI: 10.2307/2343663
[5] WERBOS P. Advanced Forecasting Methods for Global Crisis Warning and Models of Intelligence [J]. General System Yearbook, 1977, 22: 25–38
[6] SUTTON R. Learning to Predict by the Methods of Temporal Differences [J]. Machine Learning, 1988, 3(1): 9–44. DOI: 10.1007/BF00115009
[7] WATKINS C J C H, DAYAN P. Q-Learning [J]. Machine Learning, 1992, 8: 279–292
[8] RUMMERY G A, NIRANJAN, M. On-Line Q-Learning Using Connectionist Systems [M]. Cambridge, England: the University of Cambridge, 1994
[9] THRUN S. Monte Carlo POMDPS [C]//Advances in Neural Information Process-

ing Systems. Cambridge, USA, 2000
[10] SUTTON R S, MCALLESTER D A, SINGH S P, et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation [C]//Advances in neural information processing systems. Cambridge, USA, 2000
[11] SILVER D, LEVER G, HEESS N, et al. Deterministic Policy Gradient Algorithms [C]//31th International Conference on Machine Learning. Beijing, China, 2014
[12] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human - Level Control Through Deep Reinforcement Learning [J]. Nature, 2015, 518(7540): 529. DOI: 10.1038/nature14236
[13] VAN HASSELT H, GUEZ A, SILVER D. Deep Reinforcement Learning with Double Q-Learning [C]//13th AAAI Conference on Artificial Intelligence, Phoenix, USA, 2016
[14] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized Experience Replay [DB/OL]. (2015-11-18). https://arxiv.org/abs/1511.05952
[15] WANG Z, SCHAUL T, HESSEL M, et al. Dueling Network Architectures for Deep Reinforcement Learning [C]//International Conference on Machine Learning. New York, USA, 2016. DOI: 10.1155/2018/2129393
[16] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous Control with Deep Reinforcement Learning [DB/OL]. (2015 - 09 - 09). https://arxiv. org/abs/1509.02971
[17] NAIR A, SRINIVASAN P, BLACKWELL S, et al. Massively Parallel Methods for Deep Reinforcement Learning [DB/OL]. (2015-07-15). https://arxiv.org/abs/1507.04296
[18] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust Region Policy Optimization [C]//31th International Conference on Machine Learning. Lille, France, 2015
[19] 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. DOI: 10.13140/RG.2.2.18893.74727
[20] DUAN Y, CHEN X, HOUTHOOFT R, et al. Benchmarking Deep Reinforcement Learning for Continuous Control [C]//33rd International Conference on Machine Learning, New York, USA, 2016
[21] BEATTIE C, LEIBO J Z, TEPLYASHIN D, et al. Deepmind Lab [DB/OL]. (2016-12-12). https://arxiv.org/abs/1612.03801
[22] WU B. Hierarchical Macro Strategy Model for Moba Game AI [C]//AAAI Conference on Artificial Intelligence. Honolulu, USA, 2019. DOI: 10.1609/aaai.v33i01.33011206
[23] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal Policy Optimization Algorithms [DB/OL]. (2017-07-20). https://arxiv.org/abs/1707.06347
[24] HEESS N, SRIRAM S, LEMMON J, et al. Emergence of Locomotion Behaviours in Rich Environments [DB/OL]. (2017 - 07 - 07). https://arxiv. org/abs/1707.02286
[25] CLARK K. Neural Coreference Resolution [EB/OL]. https://cs224d. stanford. edu/reports/ClarkKevin.pdf
[26] JI Y, TAN C, MARTSCHAT S, et al. Dynamic Entity Representations in Neural Language Models [C]//2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark, 2017. DOI: 10.18653/v1/D17-1195
[27] BELLEMARE M G, NADDAF Y, VENESS J, et al. The Arcade Learning Environment: An Evaluation Platform for General Agents [J]. Journal of Artificial Intelligence Research, 2013, 47: 253–279. DOI: 10.1613/jair.3912
[28] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the Game of Go Without Human Knowledge [J]. Nature, 2017, 550(7676): 354. DOI: 10.1038/nature24270
[29] FERNÁNDEZ F, VELOSO M. Probabilistic Policy Reuse in a Reinforcement Learning Agent [C]//5th International Joint Conference on Autonomous Agents and Multiagent Systems. Hakodate, Japan, 2006. DOI: 10.1145/1160633.1160762
[30] MCGOVERN A, BARTO A G. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density [C]//8th International Conference on Machine Learning. Williams College, USA, 2001.
[31] DIETTERICH T G. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition [J]. Journal of Artificial Intelligence Research, 2000, 13: 227–303.
[32] MAHADEVAN S. Proto-Value Functions: Developmental Reinforcement Learning [C]//22nd International Conference on Machine Learning. Bonn, Germany, 2005: 553–560. DOI: 10.1145/1102351.1102421
[33] MADDEN M G, HOWLEY T. Transfer of Experience Between Reinforcement Learning Environments with Progressive Difficulty [J]. Artificial Intelligence Review, 2004, 21(3-4): 375–398. DOI: 10.1023/B:AIRE.0000036264.95672.64
[34] DRIESSENS K, RAMON J, CROONENBORGHS T. Transfer Learning for Reinforcement Learning Through Goal and Policy Parametrization [C]//ICML Workshop on Structural Knowledge Transfer for Machine Learning. Pittsburgh, USA, 2006
[35] LITTMAN M L. Markov Games as a Framework for Multi-Agent Reinforcement Learning [M]. Machine Learning Proceedings 1994. Burlington, USA: Morgan Kaufmann, 1994: 157–163. DOI: 10.1016/B978-1-55860-335-6.50027-1
[36] GREENWALD A, HALL K, SERRANO R. Correlated Q-Learning [C]//20th International Conference on Machine Learning. Washington D. C., USA, 2003
[37] HU J and WELLMAN M P. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm [C]//15th International Conference on Machine Learning, Madison, USA, 1998
[38] SUNEHAG P, LEVER G, GRUSLYS A, et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning [DB/OL]. (2017-06-16). https://arxiv.org/abs/1706.05296
[39] RASHID T, SAMVELYAN M, WITT C S, et al. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning [C]//35th International Conference on Machine Learning. Stockholm, Sweden, 2018
[40] LOWE R, WU Y, TAMAR A, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments [C]//Conference on Neural Information Processing Systems (NIPS). Long Beach, USA, 2017: 6379–6390.

Biographies

DONG Shaokang (shaokangdong@gmail.com) obtained his B.S. degree from the Advanced Class of Huazhong University of Science and Technology, China in 2018. He is currently a Ph.D. student in the Department of Computer Science and Technology, Nanjing University, China. His research interests include machine learning, reinforcement learning, and multi-armed bandits.

CHEN Jiarui obtained his B.S. degree from Dongbei University of Finance and Economics, China in 2018. He is currently a master student in the Department of Computer Science and Technology, Nanjing University, China. His research interests include machine learning, multi-agent reinforcement learning, and game.

LIU Yong received a B.S degree in communication engineering from China Agricultural University, China in 2017. He is currently a master student in the Department of Computer Science and Technology, Nanjing University, China. His current research interests include reinforcement learning, multi-agent learning, and transfer learning.

BAO Tianyi is an undergraduate student currently studying in the University of Michigan, USA. She studies computer science and psychology and will receive her B.S. degree in 2020. Her current research interests include the machine learning and human-computer interaction.

GAO Yang received the Ph.D. degree in computer software and theory from the Department of Computer Science and Technology, Nanjing University, China in 2000. He is a professor with the Department of Computer Science and Technology, Nanjing University. His current research interests include artificial intelligence and machine learning. He has published over 100 papers in top international conferences and journals.