Research on Multi-agent Sparse Reward Problem

Fanxiao Meng

Computer department, Shandong Xiehe Universty, Jinan, 250107, China

631401100210@mails.cqjtu.edu.cn

Abstract. Sparse reward poses a significant challenge in deep reinforcement learning, leading to issues such as low sample utilization, slow agent convergence, and subpar performance of optimal policies. Overcoming these challenges requires tackling the complexity of sparse reward algorithms and addressing the lack of unified understanding. This paper aims to address these issues by introducing the concepts of reinforcement learning and sparse reward, as well as presenting three categories of sparse reward algorithms. Furthermore, the paper conducts an analysis and summary of three key aspects: manual labeling, hierarchical reinforcement learning, and the incorporation of intrinsic rewards. Hierarchical reinforcement learning is further divided into option-based and subgoal-based methods. The implementation principles, advantages, and disadvantages of all algorithms are thoroughly examined. In conclusion, this paper provides a comprehensive review and offers future directions for research in this field.

Keywords: Sparse reward; manually annotated; hierarchical reinforcement learning; intrinsic reward.

1. Introduction

A kind of learning method of Artificial Intelligence is called Reinforcement Learning (RL). Nowadays, RL is widely used in multi-agent field. In RL, the agent is always interacting with its environment, gets extrinsic rewards from the environment and optimizes policies, constantly learns and tries and mistakes, and finally obtains the optimal policy with the maximum cumulative reward.

Traditional RL methods can only deal with simple decision problems, but the emergence of deep learning breaks this limitation [1]. A multi-layer neural network is constructed through the pre-training of the autoencoder, and the initial deep learning is generated by learning through the backpropagation algorithm. Deep Learning relies on deep neural networks to solve many of the industry's difficulties. A combination of RL and Deep Learning is called Deep Reinforcement Learning (DRL). In DRL, the processing ability of high dimensional data can be improved by using deep neural networks [2]. Combining Deep Learning with RL can improve the "insight" of AI.

As the environment becomes more complex, a single agent cannot effectively complete the task, so multiple agents are developed. Since the multi-agent system was proposed in the 1970s, it plays an important role in all walks of life. Based on the multi-agent system, the researchers introduced multi-agent reinforcement learning [3]. One of the main challenges facing RL algorithms is the sparse reward problem. Studying the solution of sparse reward problem is conducive to improving the utilization rate of algorithm samples and improving the performance of optimal strategy, speeding up the convergence rate of algorithm, and reducing the waste of computing resources. At present, researchers in various countries have done a lot of research on solving sparse reward problem, and have obtained a lot of remarkable results.

At present, most articles focus on a solution or an algorithm, but there are few review articles on sparse reward problem. This paper discusses and analyzes the solution of sparse reward algorithm from three categories: manual labeling, hierarchical RL, and introduction of intrinsic reward, and analyzes the advantages and disadvantages of each category, so as to help future researchers to have a more comprehensive understanding of sparse reward problem.

2. Sparse Reward

At present, one of the important challenges of multi-agent reinforcement learning is the sparse reward problem. In RL, the agent can only judge whether the current policy is appropriate through

the reward, and further optimize the strategy according to the reward. In reality, however, rewards are often sparse or missing, and the agent does not receive the reward after each step, only when the task is completed. The lack of reward signal leads to slow learning of the agent, which is not conducive to algorithm convergence. For example, in the game of Go, only after the game is over, the agent can receive the reward and learn about the feedback of the game, and it is difficult to determine the reward during the game. In the navigation task, only when the agent reaches the specified position within the specified time step can the agent get the corresponding reward, and the relevant reward is missing during the navigation. In the grasping task of the robot arm, the agent can only get the reward after successfully grasping the target, and if there is a mistake in the middle step, it will never get the effective reward [4].

Of all the ways to solve the problem of sparse rewards, the most straightforward is to artificially design rewards. Because artificially designed rewards are difficult to handle complex tasks, researchers have developed hierarchical reinforcement learning methods. In the field of multitasking learning, researchers have found that problems can be solved by setting up additional auxiliary tasks. When faced with a lack of prior knowledge, researchers have developed an intrinsically motivated approach to problem solving, that is, to construct an intrinsic reward for the agent.

3. Solution

3.1. Manually Annotated

The most direct way to solve the sparse reward problem is to set the reward manually [5]. According to the prior knowledge of some specific problems, the developer constructs corresponding reward functions to help the agent better complete the task.

In RL, the most basic reward function is: $R' = R + F$ where R is the reward function of the original problem, F is the additional shaping reward function, and F gives the agent an additional reward when the agent overperforms the state. Ng et al. developed Potential-based reward shaping [6]. When an agent transitions between states, corresponding rewards are added according to the difference in value of any Potential function of these states. Moreover, Ng et al. demonstrated that based on potential shaping rewards $\Upsilon\Phi(s') - \Phi(s)$, in which the optimal strategy remains unchanged.

Subsequently, Harutyunyan et al. put forward dynamic advice potentials on the basis of Potential based reward shaping [7]. According to an arbitrary reward function, this reward function can be transformed into a dynamic form on the premise that the strategy remains unchanged, and train the agent. First, the potential-based suggestion framework is extended to dynamic potential-based suggestions, and then a second-order value function with any variant of the reward function is learned in parallel, and the continuous estimation of the function is used as the dynamic advice potentials. Compared with the base agent, dynamic (PB) VF advice has a faster learning speed, and the learning performance does not degrade with the increase of time. dynamic (PB) VF advice can also quickly learn to balance poles and get a smaller variance.

Demir et al. showed that in partially observable situations, landmarks can set rewards in RL with hidden states [8]. Landmarks can be used as an influencing factor in areas such as path planning and robotic navigation. Demir et al. generate an abstract model based on the tag of a problem with a hidden state, calculate the shaping reward when the agent completes an abstract transition between two landmarks, and apply the reward to the underlying RL algorithm. Figure 1 shows Landmark Based Reward Shaping (LBRS). Compared to the basic SarsaLandmark, the SarsaLandmark with LBRS has a higher learning speed, improves the performance of the algorithm, and is able to help the agent find a better policy to complete the task faster.

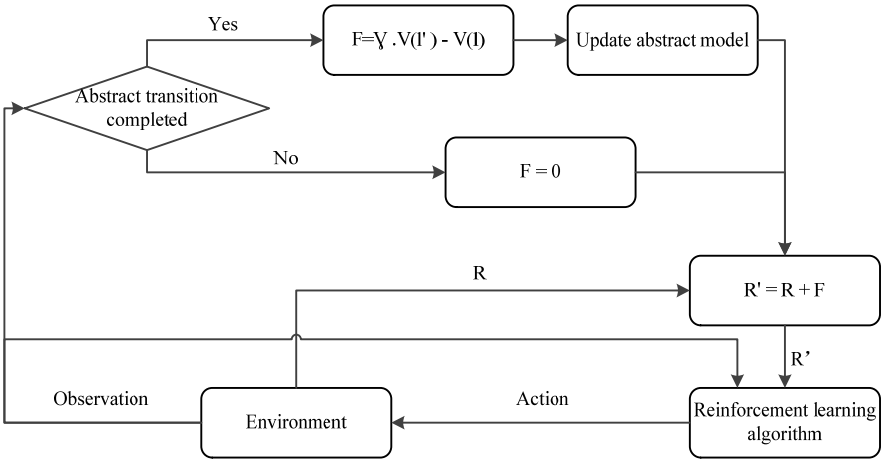

Fig. 1. LBRS workflow, combined with the underlying RL algorithm (original)

Artificially designed reward method has limitations to solve practical problems. Because the prior conditions are related to specific problems, the designed reward function lacks generality, and a new reward function must be manually set for each environment. Secondly, if the reward function design is not accurate enough, the agent will fall into the local optimal condition.

3.2. Hierarchical Reinforcement Learning

Introduce hierarchical structure to RL. Hierarchical Learning uses the idea of divide and conquer to decompose complex RL problems into multiple simple sub-problems and then solve them one by one, thus reducing the training difficulty of the agent and improving the solving efficiency of the whole task. Hierarchical learning algorithms are mostly multi-layer structures, with the upper layer controlling the lower layer, and then the lowest layer interacting with the environment. Because Hierarchical Reinforcement Learning (HRL) abstracts from time, Hierarchical Learning is not limited to policies at one point in time, but involves policies that persist over a period of time. And HRL has reuse because of the characteristics of Hierarchical Learning.

Two general categories can be used to classify HRL algorithms. One is an option-based approach. Among multiple lower-level policies, one of them is chosen by higher-level policy, and the selected lower-level policy executes the action, as shown in Figure 2. The second method is based on subgoals. Determine the lower-level subgoals based on the upper-level policy, and the lower-level policy executes actions to complete the subgoals. The managers of each layer only interact with the adjacent upper and lower layers, as shown in Figure 3.

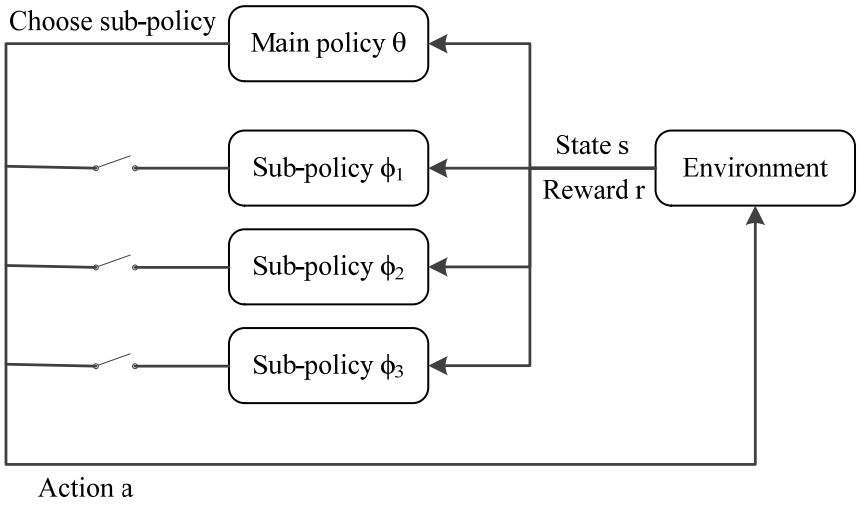

Fig. 2. Schematic of option-based HRL methods(original)

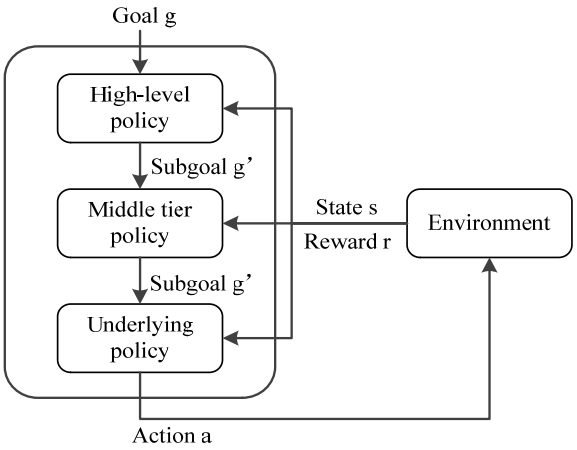

Fig. 3. Schematic of subgoal-based HRL methods(original)

Based on the hierarchical method of options, Bacon et al proposed the Option-Critic (OC) algorithm, which can learn options in a single task without affecting the learning speed [9]. The OC algorithm only needs to select the number of required options, does not need to adjust sub-goals, additional rewards and other indicators, and realizes end-to-end learning. Figure 4 displays the OC algorithm's structure diagram. The OC algorithm has strong generalization ability, and when the target changes, the option-critic agent has a fast recovery speed. However, the training efficiency of OC algorithm is low, and it is not good at dealing with complex problems.

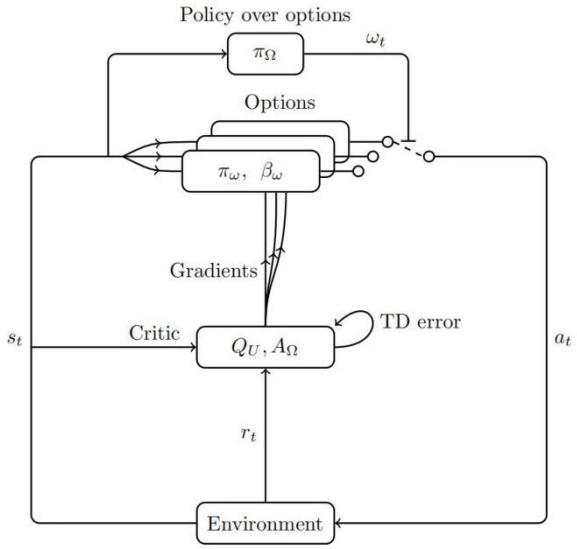

Fig. 4. Option-Critic structure diagram [4]

Gregor et al. proposed the Variational Intrinsic Control (VIC) algorithm, which introduced the idea of empowerment to discover the set of intrinsic options available to the agent [10]. Gregor et al. show that the VIC algorithm is at least suitable for classical RL cases where external rewards are maximized and where the agent has a maximum enabling target state. VIC algorithm is a closed-loop strategy that pushes agents to explore the environment.

Based on the hierarchical method of subgoals, Vezhnevets et al developed a framework called Feudal Networks (FuN), in which the subgoals were configured as the variation direction of the potential state space [11]. During operation, the agent could automatically learn the subgoals vector. The Manager layer and the Worker layer are the two levels that make up the FuN framework (Figure 5). The Manager sets the goal at a lower time resolution based on the learning signal from the external environment; Based on the learning signal from within the system, the Worker performs the action and completes the target at a higher time resolution. The overall running process of the FuN framework is roughly as follows: first, feature extraction, then sub-target generation, then Worker action strategy generation, and finally network structure training. The structure diagram of the FuN

framework is shown below. The FuN algorithm does not require prior environmental knowledge, improves long-term credit allocation and memory organization, and increases the lifetime of cyclic state memory. In the Montezuma Revenge game experiment, FuN was able to start learning earlier and get higher scores.

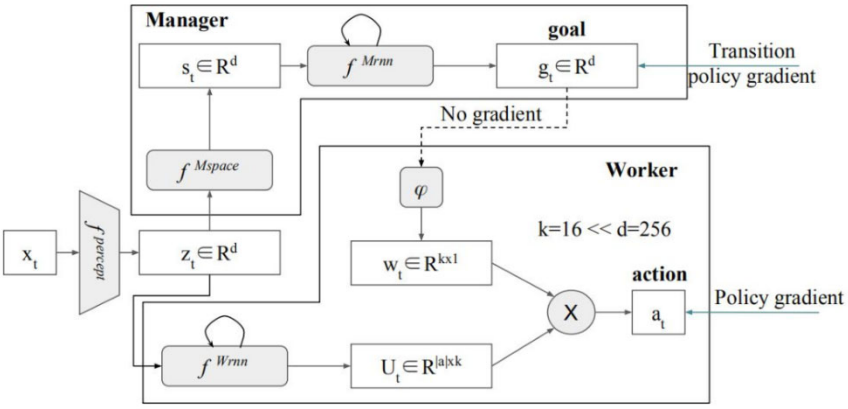

Fig. 5. FuN structure diagram [6]

Levy et al. proposed Hierarchical Actor Critic (HAC) algorithm [12]. The HAC algorithm increases the learning speed by allowing agents to learn strategies at multiple levels simultaneously. The specific hierarchical architecture and the method of parallel learning multi-level strategies constitute the framework of HAC algorithm. The HAC algorithm can overcome the instability of the training policy of the agent on different time scales.

Table 1. Algorithm analysis and comparison

Classical algorithm	Algorithm type	Innovation point	Drawback
OC	Synchronous option - deep hierarchical reinforcement learning (SO-DHRL)	The O-DHRL framework is constructed for the first time, and the function is optimized by the strategy gradient method to increase the extensibility.	The stratification problem is significant; The option precision is insufficient and cannot be accurately distinguished.
VIC	Asynchronous option - deep hierarchical reinforcement learning (AO-DHRL)	For the first time, using maximum empowerment on top of DHRL allows you to learn skills that are less relevant to the task.	There are hierarchical problems; Fewer skills are available.
FuN	Foresight subgoal - deep hierarchical reinforcement learning (FG-DHRL)	Guided subtargets are introduced for the first time, and a two-layer model is constructed, which has the characteristics of end-to-end, differentiable and modular. The algorithm is flexible and can handle continuous spatial tasks.	The module structure is complex; The performance index changes greatly; It is difficult to adjust the parameters, which easily leads to the collapse of the model.
HAC	Hindsight subgoal - deep hierarchical reinforcement learning (HG-DHRL)	HER algorithm is extended to multi-layer structure, which solves the problem of agent learning instability and has high scalability.	Convergence unknown.

As can be seen from the Table 1, with the development of O-DHRL algorithm and G-DHRL algorithm, the performance of reinforcement hierarchical learning has been continuously improved, the amount of prior experience required has been reduced, the transferability has been enhanced, the sub-goal completion rate has been increased, and the data utilization has been improved [13].

3.3. Introduce Intrinsic Reward

When the agent is in the state of lack of external reward, intrinsic reward can play a key auxiliary role. Intrinsic rewards are conducive to driving agents to explore a larger and wider range of environments. After the intrinsic reward mechanism is introduced into the RL algorithm, the task execution of the agent is not only based on the reward provided by the external environment, but also based on the intrinsic reward provided by the agent itself. The concrete RL framework is shown in Figure 6. The introduction of intrinsic rewards can increase the exploration efficiency and training speed of the agent.

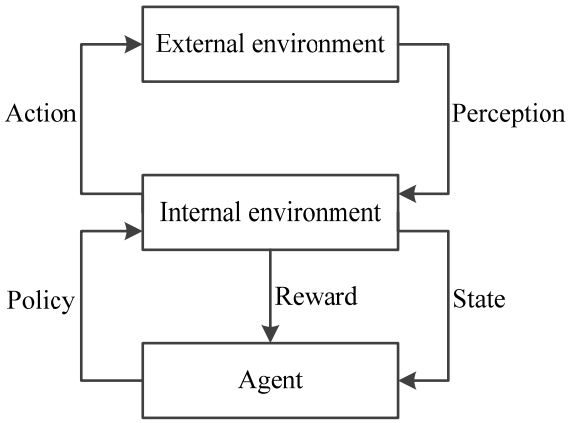

Fig. 6. Intrinsic motivation RL framework (original)

Based on the reward function of Markov Decision Process, Houthooft et al. explored an exploration strategy based on maximizing the information benefits and rewards of agents to the environment [14]. This strategy is called Variational Information Maximizing Exploration (VIME). VIME is a kind of Bayesian neural network and variational inference, which is used to maximize the information gain of the environment dynamics, which is conducive to the exploration of the agent in the uncertain environment, and can solve the problem of poor effectiveness and scalability of the exploration method in the high-dimensional complex environment.

Based on the modeling difficulties caused by the randomness of the environment, Pathak et al. developed the Intrinsic Curiosity Module (ICM) [15]. ICM predicts the next state based on the current state and action, calculates the error from the actual state, and then uses this prediction error as an intrinsic reward and encourages the agent to explore the unfamiliar environment. The calculation of intrinsic rewards is shown in Figure 7. ICM is the first way for an agent to learn to navigate a 3D environment from pixels without any external rewards.

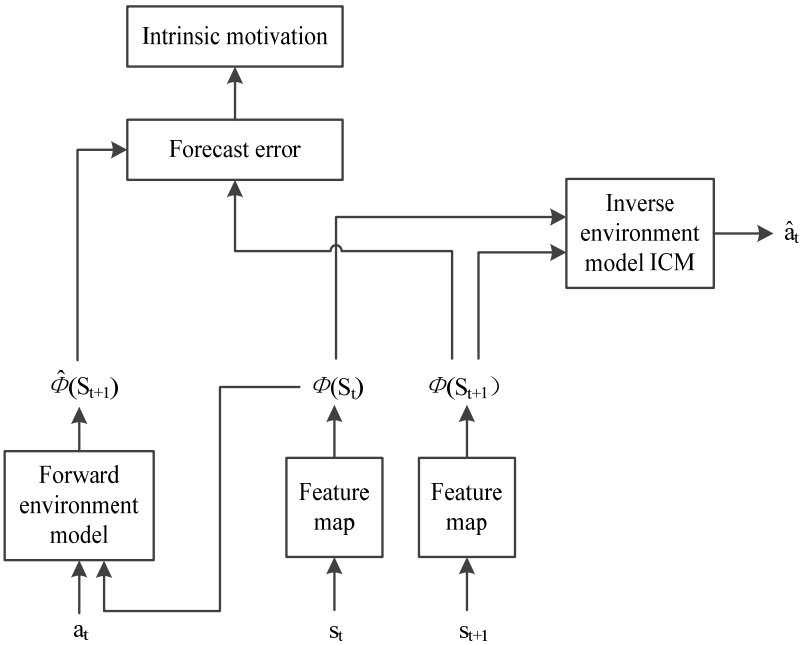

Fig. 7. Intrinsic motivation computed by inverse environment model [15]

On the basis of large-scale research on pure curiosity-driven learning, Burda et al. studied random network distillation (RND), which transforms the environment into a random network [16]. RND judges the sample distribution distance between the two networks, measures whether the measured state is novel, and gives the agent the corresponding intrinsic reward based on the above judgment. RND can motivate agents to explore unfamiliar environments. In game experiments, RND enables the agent to complete the game without using a demonstration or knowing the basic state of the game.

Since most novelty estimation methods build models based on explicit current or next state distributions, models are difficult to train in complex, high-dimensional environments. Fu et al. developed the EX^2 method to classify and judge novelty [17]. The EX^2 classifier can distinguish novelty states from other states and thus estimate novelty based on how difficult the distinction is. In simple environments, EX^2 can provide density estimates of potential states without generating any explicit training. In complex environmental experiments, the implicit density estimation generated by EX^2 provides exploration rewards as good as those provided by various explicit density estimation techniques. EX^2 has good practicality and scalability.

However, whether the intrinsic reward will converge to zero is a comprehensive problem, which includes many factors such as environmental randomness and model validity. Since the aim of the intrinsic incentive is to encourage the agent to investigate the environment, the environment is similar to the black box state for the agent, so after an intrinsic reward is used, it is not appropriate to delete the intrinsic reward directly under any circumstances.

4. Conclusion

Through research, this paper finds that the sparse reward problem is studied and solved by researchers through various aspects at the same time. For manual design reward value, it is the most direct, but it is too targeted, and the workload is large, and it requires a lot of professional knowledge in the relevant direction. For hierarchical reinforcement learning, it has high stability, convergence speed and learning efficiency, but it still needs a lot of artificial design rewards and lacks universal applicability. For the introduction of intrinsic rewards, data efficiency and generalization ability are improved, but when the agent is unable to complete the task in the first place, the strategy is not effectively trained.

This paper mainly introduces the algorithm to solve sparse reward based on three aspects: manual annotation, hierarchical reinforcement learning and the introduction of intrinsic reward, which is

conducive to relevant researchers to understand the existing solution and help relevant personnel to improve and innovate on the existing basis.

At present, reinforcement learning methods still lack human ability to think, reason and make decisions independently. With the development of technology, algorithms can be applied to more and more complex problems and can be more convenient for People's Daily life.

Reference

[1] Zhao Ying. Research on intrinsic rewards for reinforcement learning [D]. Guizhou University, 2022.

[2] Yin Jia. Research and implementation of sparse reward task solving method in reinforcement learning [D]. University of Electronic Science and Technology of China, 2022.

[3] Du Wei, Ding Shi-fei. Overview on Multi-agent Reinforcement Learning [J]. Computer Science, 2019, 46(08): 1-8.

[4] Ma Yun-ting. Research on reward mechanism of multi-agent reinforcement learning [D]. Hefei University of Technology, 2021.

[5] Yang Wei-yi, Bai Chen-jia, Cai Chao, Zhao Ying-nan, Liu Peng, et al. Survey on Sparse Reward in Deep Reinforcement Learning [J]. Computer Science, 2020, 47(03): 182-191.

[6] Ng A. Y, Harada D, Russell S. Policy invariance under reward transformations: Theory and application to reward shaping [C]. Icml, 1999: 278-287.

[7] Harutyunyan A, Devlin S, Vrancx P, et al. Expressing arbitrary reward functions as potential-based advice [C]. Proceedings of the AAAI Conference on Artificial Intelligence, 2015: 2652- 2658.

[8] Demir A, Çilden E, Polat F. Landmark based reward shaping in reinforcement learning with hidden states [C]. Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems, 2019: 1922-1924.

[9] Bacon P L, Harb J, Precup D. The option-critic architecture [C] Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, USA, 2017.

[10] Gregor K, Rezende DJ, Wierstra D. Variational intrinsic control. arXiv, 2016: 1611.07507.

[11] Vezhnevets A S, Osindero S, Schaul T, et al. Feudal networks for hierarchical reinforcement learning [C] Proceedings of the 34th International Conference on Machine Learning-Volume 70. Sydney, Australia, 2017: 3540−3549.

[12] Levy A, Konidaris G, Platt R, et al. Learning multi-level hierarchies with hindsight [J]. arXiv preprint arXiv, 2017: 1712.00948.

[13] Huang ZG, Liu Q, Zhang LH, Cao JQ, Zhu F. Research and Development on Deep Hierarchical Reinforcement Learning. Ruan Jian Xue Bao/ Journal of Software, 2023, 34(2): 733 760 (in Chinese).

[14] Houthooft R, Chen X, Duan Y, et al. Vime: Variational information maximizing exploration [C] Advances in Neural Information Processing Systems. 2016: 1109-1117.

[15] Pathak D, Agrawal P, Efros A A, et al. Curiosity-driven exploration by self-supervised prediction [C] International conference on machine learning. PMLR, 2017: 2778-2787.

[16] Burda Y., Edwards H., Storkey A, Klimov O. Exploration by random network distillation [J]. arXiv preprint arXiv, 2018: 1810.12894.

[17] Fu J, Co-Reyes J, Levine S. Ex2: Exploration with exemplar models for deep reinforcement learning [C] Advances in Neural Information Processing Systems. 2017: 2577-2587.