Information Sciences 662 (2024) 120255

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Explorer-Actor-Critic: Better actors for deep reinforcement learning

Junwei Zhang [a,b], Shuai Han [a,b,c], Xi Xiong [a,b], Sheng Zhu [a,d], Shuai Lü [a,b,d,*]

[a] Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China
[b] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[c] Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands
[d] College of Software, Jilin University, Changchun 130012, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Sample efficiency
Overestimation bias
Actor-critic framework
Exploration and exploitation

ABSTRACT

Actor-critic deep reinforcement learning methods have demonstrated significant performance in many challenging decision-making and control tasks, but also suffer from high sample complexity and overestimation bias. Current researches focus on using underestimation to balance overestimation and reducing bias through ensemble learning, but introducing underestimation bias and excessive network costs. In this paper, we first analyze the effect of action selection policy on estimation bias. Then, we propose the Explorer-Actor-Critic (EAC) method that gives a more conservative objective for the actor to reduce overestimation, introduces a learnable explorer to improve exploration ability, and uses an action mixing mechanism to mitigate experience distribution bias. Furthermore, we apply the EAC method to TD3 and SAC and verify its effectiveness through extensive comparison and ablation experiments. Our algorithm not only outperforms state-of-the-art algorithms, but also is compatible with other actor-critic methods.

1. Introduction

Model-free deep reinforcement learning algorithms have yielded significant performance in a range of challenging domains, such as arcade games [1,2] and robotic control [3]. Actor-critic methods are the most popular methods in continuous domains of reinforcement learning. These methods provide gradients for policies by estimating value functions. Deep Deterministic Policy Gradient (DDPG) [4] is one of the most classical actor-critic methods. However, it suffers from severe overestimation bias [5]. This phenomenon arises from the training of the value function, in which estimation errors are propagated forward and accumulated because of the maximum operation in the Bellman equation. How to mitigate the negative effects of overestimation bias has become an essential issue.

The approaches to address the overestimation bias in the actor-critic methods mainly originate from Double Deep Q-learning [6], which counteracts the overestimation bias by changing the objective optimization of the value function. Twin Delayed Deep Deterministic Policy Gradient (TD3) [5] uses the minimum of two value function networks as the value estimation, mitigating the effects of overestimation with a small amount of underestimations, which is an outstanding method with significant results currently. But these underestimations can also reduce exploration efficiency, especially for the rarely visited states [7]. Randomized Ensembled

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun 130012, China.
E-mail addresses: junwei20@mails.jlu.edu.cn (J. Zhang), s.han@uu.nl (S. Han), xiongxi21@mails.jlu.edu.cn (X. Xiong), zhusheng20@mails.jlu.edu.cn (S. Zhu), lus@jlu.edu.cn (S. Lü).

https://doi.org/10.1016/j.ins.2024.120255
Received 28 March 2023; Received in revised form 26 January 2024; Accepted 26 January 2024
Available online 1 February 2024

Double Q-Learning (REDQ) [8] uses multiple value function networks to reduce the overestimation bias based on ensemble learning, which improves the utilization of experience while requiring more complex network structures and training times than other methods.

In summary, the current algorithms for mitigating overestimation bias can be categorized into two types: 1) modifying the objective of the critic to mitigate the overestimation bias; 2) introducing ensemble learning method to reduce bias. However, modifying the objective of critic can not simultaneously consider the exploration and exploitation [7]. Ensemble-based methods need more networks or more training times [8].

We find that the action policy of the actor needs to select actions for both the critic value estimation and the environment interaction. We can optimize the actor to obtain a solution that can balance overestimation bias and exploration ability. Therefore, we investigate how to train better actors to improve the accuracy of the value function and training efficiency. In actor-critic methods, we can only expect the actor network to find the maximum of the value function which is subject to training errors and noises. Only maximizing a single value function estimated in existing methods is not the best choice. The impact of bias and potential exploration values should also be considered.

In this paper, we consider the effects of the action policy in actor-critic methods and divide the actor into two parts: computing the estimation in the value function optimization process and finding the appropriate action in the environmental interaction process. Among them, in the value function optimization process, we replace the actions that have larger errors with more appropriate actions to mitigate the accumulation of overestimation bias. In the environmental interaction process, the actor needs to focus on accessing unknown states or unknown actions while maximizing the value function.

We propose a new method called Explorer-Actor-Critic (EAC), which divides the original actor into an actor and an explorer optimized with different objectives. Actor is used in the value function optimization to select actions with less estimation errors. Explorer is used in the environmental interaction process to select more valuable actions to explore.

We perform extensive experiments in continuous control benchmarks from OpenAI Gym, where we compare our method against the current state-of-art methods including TD3 and SAC. The results show that EAC outperforms state-of-the-art algorithms because of the mitigation of overestimation errors and the assurance of efficient exploration.

The main contributions of this paper are as follows:

- We propose the EAC method, which can train the value function more stably while ensuring the exploration ability for the agent.
- We confirm the existence of more conservative policies that can mitigate overestimation bias, and we propose an actor objective for estimation error control based on a dual value function.
- We propose an exploration policy that can select more worthwhile actions for exploration.
- We propose an action mixing mechanism to reduce the negative impact of experience distribution bias.
- We experimentally verify that EAC method can improve the training efficiency and be applied to various actor-critic methods.

The remaining sections are organized as follows. Section 2 details the related works about overestimation bias, exploration ability and experience distribution bias. Section 3 gives the theoretical basis and symbol representation. Section 4 proposes the EAC method, which provides a three-part optimization that can improve exploration and exploitation. Section 5 conducts various comparative experiments and analyses the effect of the EAC method. Section 6 summarizes conclusions and future work.

2. Related work

Our method changes the training objectives of the actor, which is able to improve the training efficiency of reinforcement learning methods in terms of both reducing overestimation bias and improving exploration ability, and takes into account the problem of experience distribution bias from different action selection policies. Therefore, our work is related to three domains: mitigating overestimation bias, improving exploration ability, and reducing experience distribution bias.

2.1. Mitigating overestimation bias

Double Deep Q-learning [6] uses two value functions for action selection and value estimation respectively, which effectively mitigates overestimation bias but introduces underestimation bias. Weighted Double Q-learning [9] introduces a weight to balance the underestimation bias in Double Q-learning and overestimation bias in Q-learning. Averaged Q-learning [10] and Maxmin Q-learning [11] are based on ensemble learning, using multiple value networks to obtain low bias and low variance estimation by taking the average or max-min operator respectively. However, this method requires a large number of additional value networks and training times. TD3 [5] follows the idea of Double Deep Q-learning by taking the minimum of two value functions to reduce the overestimation error. REDQ [8] is based on the Maxmin Q-learning, which uses multiple value networks to reduce the approximation error and variance of the estimation. In addition, the inconsistency between the actor network and critic network is also one of the reasons for the overestimation bias. Soft Actor Critic (SAC) [12] makes the two distributions more approximate by introducing entropy values. Softmax Deep Double Deterministic Policy Gradients (SD3) [13] reduces the inconsistency by taking multiple suboptimal actions and making a probabilistic selection.

In addition, the PPO [14] algorithm can also be constructed based on the actor-critic framework, which does not suffer from the problem of overestimation bias because of “On-Policy” learning. But it has lower sample efficiency and suffers from catastrophic forgetting problem.

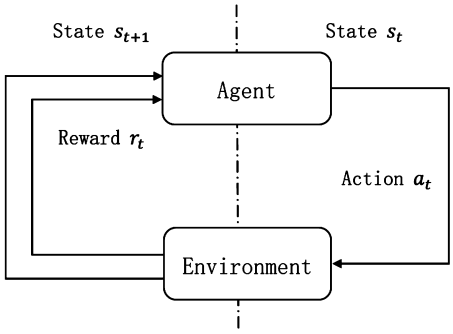


Fig. 1. The framework of reinforcement learning system.

2.2. Improving exploration ability

For high-dimensional or extremely sparse reward tasks, Intrinsic Curiosity Module [15,16] encourages the agent to visit more novel states by introducing prediction errors. Pseudo-counting methods [17,18] estimate the number of state visits to increase the exploration range. These methods find more valuable exploration directions according to the characteristics of the environments, which means these methods are only with significant effects in specific tasks in most cases. In continuous control tasks, there are also methods to analyze the action selection policy of the actor. Optimistic Actor Critic [7] considers the relationship between the gradient optimization direction and the potential exploration direction, and enhances the exploration ability by adding the guidance of the gradient direction to the action. In contrast, our method considers both the stable optimization of the value function and the exploration ability of the agent.

2.3. Reducing experience distribution bias

Estimation bias may occur when the action selection policies used for value estimation and environment interaction do not match, especially for out-of-distribution (OOD) state-action pairs, which are widely studied in offline reinforcement learning. Batch-Constrained deep Q-learning [19] finds the best action in the neighborhood of existing empirical actions by imposing constraints on the action policy. Bootstrapping Error Accumulation Reduction [20] and Behavior Regularized Actor Critic [21] use the Kullback-Leibler divergence and Maximum Mean Discrepancy respectively to penalize the gap between selected actions and empirical actions. Conservative Q-learning [22] underestimates OOD actions by regularizing the value function to reduce the probability of selecting them.

3. Preliminaries

Reinforcement learning, as one of the methods of machine learning, collects and learns experience data obtained from the interaction between the agent and the environment in order to find the optimal policy for decision making tasks.

3.1. Reinforcement learning framework

In the reinforcement learning problem, the reinforcement learning system is usually constructed by two parts of the agent and the environment [23]. The system framework is shown in Fig. 1.

As in Fig. 1, the agent can observe the current state s_t of the environment and choose the action a_t to be executed. The environment is influenced by the action, transferring the state to s_{t+1} and feeding back a reward r_t to the agent. For reinforcement learning tasks, the agent is not instructed on which action is correct, but rather optimizes its policy in a "trial and error" manner. Simultaneously, actions can often affect rewards of multiple moments, i.e., reinforcement learning also needs to solve the problem of delayed rewards.

Reinforcement learning searches for the behavior that maximizes the rewards from the interaction between agent and environment. This problem is usually formalized as a Markov Decision Process [24] with a tuple (S, A, P, R, γ), where S and A are the state space and action space, $P : S \times A \times S \to [0,\ 1]$ is the transition probability, $R : S \times A \times S \to \mathbb{R}$ is the reward function, and $\gamma \in [0,\ 1]$ is the discount factor. For each time step, the agent gets the observation $s_t \in S$ from the environment, selects an action $a_t \in A$ to interact with the environment, and then gets the reward $r_t \in R$ according to the reward function and the next state s_{t+1}. The agent needs to find an optimal policy $\pi : S \times A \to [0, 1]$, which maximizes the cumulative discounted reward $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. There are various methods to find the optimal policy π for the agent, and actor-critic framework has more significant performance as one of the widely used methods.

3.2. Actor-critic methods

The actor-critic framework uses two networks or function approximators to construct and complete the training of the agent, which is effective in tasks of continuous action control. Both the Q-based value function method and the policy gradient method

can use the actor-critic framework. The actor is the action policy network $\pi(s)$. The critic is an estimate of the current action value $Q(s,a)$ or state value $V(s)$.

3.2.1. Value function method

In the value function method, the agent directly calculates the current value function $Q_\pi(s,a)$, where Q denotes the action value corresponding to the current action and the current state. It is optimized by the Bellman residual $[Q(s_t,a_t)-Y]^2$, where $Y=r_{t+1}+\gamma\max_{a'\in A}Q^*(s_{t+1},a')$. So the current action value function $Q_\pi(s,a)$ is constantly converging to the optimal action value function $Q^*(s_{t+1},a')$. The objective of the critic is shown in Eq. (1).

$$\mathcal{J}(Q_\theta)=\mathbb{E}_{(s,a,s')\sim D}\left[(r(s,a)+\gamma Q_\theta(s',\pi_\phi(s'))-Q_\theta(s,a))^2\right] \tag{1}$$

Where, θ is the parameter of the value function network Q, ϕ is the parameter of the action policy network π, and D is the state-action domain of the environment. The goal of the action policy network is to select the action that maximizes the Q-value function in the current state. The objective of the actor is shown in Eq. (2).

$$\mathcal{J}(\pi_\phi)=\mathbb{E}_{s\sim D}\left[Q_\theta(s,\pi_\phi(s))\right] \tag{2}$$

DDPG [4], as one of the classical value function algorithms, takes the advantage of DDQN algorithm, introduces the target network to improve the stability of training, and controls the update rate of the target network by soft updating. The objective of its value function is shown in Eq. (3).

$$\mathcal{J}(Q_\theta)_{DDPG}=\mathbb{E}_{(s,a,s')\sim\mathcal{B}}\left[(r(s,a)+\gamma\hat{Q}_{\theta'}(s',\hat{\pi}_{\phi'}(s'))-Q_\theta(s,a))^2\right] \tag{3}$$

where $\hat{Q}_{\theta'}$, $\hat{\pi}_{\phi'}$ correspond to the target networks, with soft updating in the form of $\theta'=(1-\tau)\theta'+\tau\theta$, $\mathcal{B}$ is the experience replay buffer obtained by the agent interacting with the environment.

TD3 [5], as one of state-of-art algorithms, proposes minimized value function, policy delay and target noise to mitigate overestimation bias, so its value function objective is given by Eq. (4).

$$\mathcal{J}(Q_\theta)_{TD3}=\mathbb{E}_{(s,a,s')\sim\mathcal{B}}\left[(r(s,a)+\gamma\min_{i=1,2}\hat{Q}_{\theta_i'}(s',\hat{\pi}_{\phi'}(s')+\epsilon)-Q_\theta(s,a))^2\right] \tag{4}$$

where $\hat{Q}_{\theta'}$, $\hat{\pi}_{\phi'}$ correspond to the target networks, $\mathcal{B}$ is the experience replay buffer, $\epsilon\sim clip(\mathcal{N}(0,\sigma^2),-c,c)$ is random action noise obeying a clipped Gaussian distribution.

SAC [12] algorithm is based on Maximum Entropy Reinforcement Learning theory and uses probability distributions to represent action policies, replacing the original representation of combinations of Gaussian noise and deterministic action combinations. It introduces action selection probabilities into value estimation via a maximum entropy term and has better results than TD3 in some tasks, but relies more on hyperparameter tuning. Its value function and action policy objectives are shown in Eq. (5) and Eq. (6).

$$\mathcal{J}(Q_\theta)_{SAC}=\mathbb{E}_{(s,a,s')\sim\mathcal{B}}[(\gamma(\min_{i=1,2}\hat{Q}_{\theta_i'}(s',\pi_\phi(s'))-\alpha\log P(\pi_\phi(s')))+r(s,a)-Q_\theta(s,a))^2] \tag{5}$$

$$\mathcal{J}(\pi_\phi)_{SAC}=\mathbb{E}_{s\sim D}\left[Q_\theta(s,\pi_\phi(s))-\alpha\log P(\pi_\phi(s))\right] \tag{6}$$

where, $\log P(\pi_\phi(s))$ is an approximation of the entropy and the action policy is denoted by $\mathcal{N}(\mu,\sigma^2)$.

3.2.2. Policy gradient method

In the policy gradient method, it is necessary to use the current action policy network, which interacts with the environment and obtains the trajectory data η. The agent calculates the gradient of the policy and adjusts the parameters, which can increase the probability of selecting a higher value action and continuously optimize towards a better policy.

The critic is the state value function $V(s)$, where V denotes the average state value corresponding to the current state. The objective of the state value is shown in Eq. (7).

$$\mathcal{J}(V_\theta)=\mathbb{E}_{s_t\in\eta,\eta\sim\pi_\phi}[(R(s_t)-V_\theta(s_t))^2] \tag{7}$$

where, $R(s_t)=\sum_{k=t}^{n}\gamma^{k-t}r_k$ is the cumulative reward of the state s_t, η is obtained by current policy π_ϕ sampling in the environment. The actor adjusts the action selection probability in terms of the state value, and its objective is shown in Eq. (8).

$$\mathcal{J}(\pi_\phi)=\mathbb{E}_{s_t\in\eta,\eta\sim\pi_\phi}\left[(R(s_t)-V_\theta(s_t))\,log P(\pi_\phi(s))\right] \tag{8}$$

PPO [14] algorithm allows the policy gradient algorithm to reuse the experience data, uses importance sampling to adjust the distribution of action-states between the current policy and the experience, and introduces ratio clipping to control the difference between the current and the old action policies. Its policy objective is shown in Eq. (9).

$$\mathcal{J}(\pi_\phi)_{PPO}=\mathbb{E}_t\left[min(r_t(\phi)A_t,clip(\epsilon,r_t(\phi))A_t)\right] \tag{9}$$

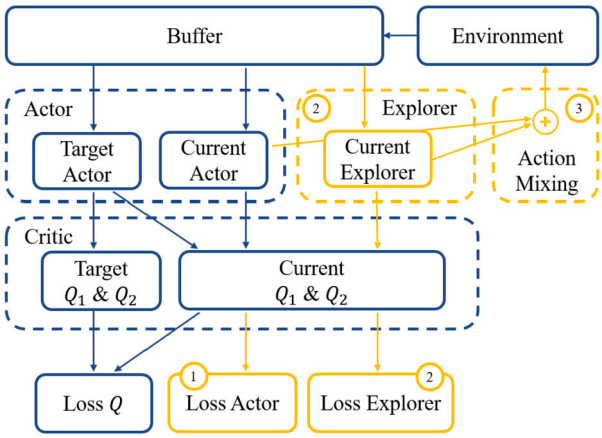


Fig. 2. The structure of EAC method. The arrows indicate the flow of data, the yellow parts are the improvements.

where, $A_t = R\left(s_t\right) - V_\theta\left(s_t\right)$ is the action advantage value, and importance sampling rate $r_t(\phi) = logP(\pi_\phi(s))/logP(\pi_{\phi_{old}}(s))$ represents action selection probability ratio of new and old policies. The $clip$ function removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$.

4. Explorer-actor-critic

Our method focuses on how to learn better actors in order to reduce overestimation bias generated from the value function in the optimization process, while maintaining a better exploration ability.

4.1. Structure of the EAC method

The actor-critic method uses an actor to give the action selection policy at the input state. The actor is applied in both the value function optimization and environment interaction sampling phase with the objective of maximizing action value, which is theoretically to converge to the optimal policy. However, in real tasks, affected by the estimation error and the sampling limitation, the actor-critic method often faces two problems:

1) Overestimation. In the training process, overestimation due to the fact that random errors in value estimation will be more likely to propagate forward based on Bellman residual. Meanwhile, the introduction of underestimation to mitigate overestimation leads to underestimation bias.

2) Exploring inefficiencies. In the environmental interaction process, the actor needs to visit more novel states for exploration, and this also increases the risk of overestimation. So the actor often needs to maintain the stability of the policy at the expense of the ability to explore.

Existing methods that only modify the objective of critic value estimation are difficult to balance both overestimation bias and exploratory ability. And the objective of the actor is not comprehensive. So we propose the Explorer-Actor-Critic (EAC) method to separate the two functions of the actor, using two action policy networks as explorer and actor, which achieve the two-part roles of value function optimization and environmental interaction sampling respectively. They give more accurate valuations in value function optimization and select more exploratory actions during environmental interactions, and the method structure is shown in Fig. 2.

In Fig. 2, we use yellow to mark the parts that differ from the baseline method, including the training objective of the actor, the introduction of the explorer network, and the use of action mixing. We clearly distinguish value networks by 'target' and 'current', where the target network is obtained by soft update of the current network. The target network is mainly involved in the computation of Bellman residual, while the actor and explorer objectives are computed by the current network.

There are three key elements in our improvement:

1) Optimization of training objectives for actor. We give a conservative actor action selection policy, which gives an optimization scheme which uses double value functions with variance control to reduce the generation and propagation of the overestimation bias.

2) Introducing the explorer network. We use explorer to give more potentially valuable actions to explore during the environmental interaction process, motivating the agent to explore less-visited novel states, which can provide the agent stronger exploratory capabilities without interfering with the training of the value function.

3) Using Action Mixing. We use action mixing to constrain policy differences between the actor and the explorer, maintaining the correlation between the distribution of environmentally interacting actions and experience pool actions.

Therefore, our descriptions are organized around the schemes and effects for each of the elements in the EAC framework.

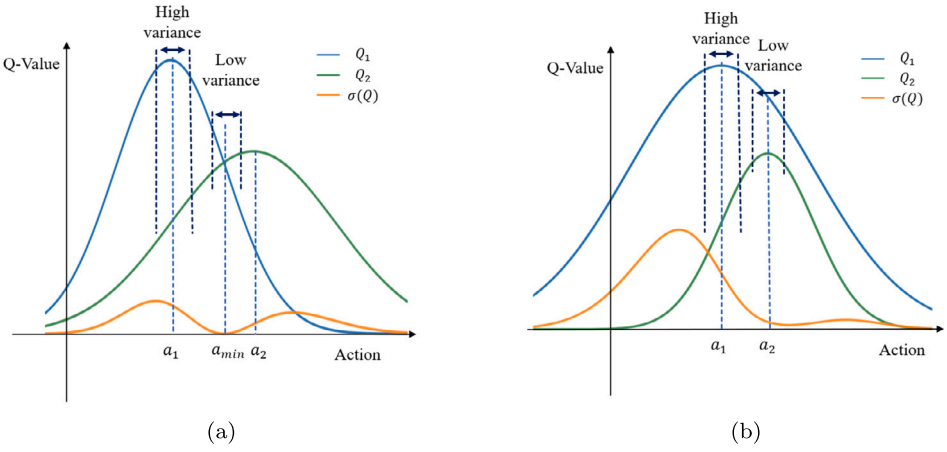


Fig. 3. Two cases of estimated value and standard deviation distributions of two value functions Q_1 and Q_2. (a) is the case of $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) < Q_2(s,a_2)$, and (b) is the case of $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) > Q_2(s,a_2)$. In addition Q_1, Q_2, a_1 and a_2 have four possible cases in total, the other two are consistent with (a) and (b).

4.2. Estimation error and overestimation bias

In this subsection, we give a definition of estimation bias, analyze the reasons why it arises, and clarify the problems faced by the off-policy methods.

Almost all off-policy methods guarantee an unbiased estimation of the value function. However, the value function estimates are randomly perturbed due to system noises. This estimation error accumulates with the maximum operation in the Bellman residual, as the overestimation bias is more likely to propagate forward than the underestimation bias. Theoretically, this is because $\mathbb{E}[\max_{a'\in A} Q(s_{t+1},a')] \geq \max_{a'\in A} \mathbb{E}[Q(s_{t+1},a')] = \max_{a'\in A} Q^*(s_{t+1},a')$, where $Q(s_{t+1},a') = Q^*(s_{t+1},a') + e_{a'}$ is the unbiased estimate of the true value, $e_{a'}$ is the random error with zero mean. So the estimation bias Δ_{sa} for each update is

$$\begin{aligned}\Delta_{sa} &= \mathbb{E}_{e_{a'}}[\tilde{Y} - Y] \\ &= \gamma \mathbb{E}_{e_{a'}}[\mathbb{E}_{s_{t+1}\sim\mathcal{B}}[\max_{a'} Q(s_{t+1},a')]] - \mathbb{E}_{s_{t+1}\sim\mathcal{B}}[\max_{a'} Q^*(s_{t+1},a')]] \\ &= \gamma \mathbb{E}_{s_{t+1}\sim\mathcal{B}}[\mathbb{E}_{e_{a'}}[\max_{a'}(Q^*(s_{t+1},a') + e_{a'})] - \max_{a'} Q^*(s_{t+1},a')]\end{aligned} \tag{10}$$

Since $\mathbb{E}_{e_{a'}}\left[\max_{a'}\left(Q\left(s',a'\right) + e_{a'}\right)\right] \geq \max_{a'} Q\left(s',a'\right)$, so there is $\Delta_{sa} \geq 0$, i.e., subject to the estimation error $e_{a'}$, estimation bias exists and is relayed forward with the optimization process.

We can use a uniform distribution $e_{a'} \sim U(-\tau,\tau)$ [25] to approximate the random error. Assuming the number of samples for the experience of replay buffer is n_{sa}, then $\tau \propto \sqrt{3\omega^2/n_{sa}}$ where ω^2 is the sample variance of replay buffer for (s,a) [11]. The range of random error τ decreases as the number of samples n_{sa} increases, so we consider a high-value action to be good enough when the random error is reduced to small enough.

4.3. Actor for reducing estimation error

In this subsection, we detail the method that can mitigate estimation errors. This method reduces the generation and propagation of estimation errors by optimizing the objective of the actor.

It has become a common approach to use the minimum of multiple value functions to mitigate overestimation bias, and usually the use of two value functions can effectively improve the performance while maintaining training efficiency, such as TD3 and SAC. However, the objective of the actor is only to maximize a single value function, which is given by

$$\mathcal{J}(\pi_\phi)_{TD3} = \mathbb{E}_{s\sim\mathcal{B}}[Q_1(s,\pi_\phi(s))] \tag{11}$$

In the optimization process, estimation error is related to the actions due to the differences in the number of interaction and training times. The objective of the actor is not limited to the highest estimated value and ignored the estimation error. We select an appropriate action policy taking into account the distribution of estimation errors in double value functions.

Although multiple value networks may give different estimated values, their training objectives are the same. Since the value function is continuous, the estimated values of two value networks divided into two cases are shown in Fig. 3. We define $a_1 = \arg\max_{a\in A} Q_1(s,a)$ to denote the action that makes Q_1 take the maximum value, and $a_2 = \arg\max_{a\in A} Q_2(s,a)$ corresponds to Q_2.

Theoretically, multiple value networks have the same training objective, and each network gives an estimate of the true Q value. So the expectation of the Q-network estimates is the same and equal to the true action value. However, each estimation error can

propagate forward with the Bellman residual resulting in the estimation bias $\Delta_{sa} \geq 0$. Meanwhile, we can get that there is a positive correlation between the estimation error, the variance between multiple Q-values, and the estimation bias, i.e., $\tau \propto \sigma(s,a) \propto \Delta_{sa}$.

Property 1. *Assuming the value function $Q(s,a)$ is continuous, there exists an action selection policy that has a smaller estimation error than the objective of maximizing a single value function while guaranteeing the convergence of reinforcement learning training. i.e. There exists a $\pi'_\phi(s)$ that satisfies selecting actions with Q values no less than the original policy: $\mathbb{E}[Q(s,\pi'_\phi(s))] = \mathbb{E}[Q(s,\arg\max_{a\in A}Q_1(s,a))] = \mathbb{E}[Q(s,\arg\max_{a\in A}Q_2(s,a))]$, and the policy $\pi'_\phi(s)$ has smaller variance: $\mathbb{E}[\sigma(s,\pi'_\phi(s))] \leq \min_{i=1,2}\mathbb{E}[\sigma(s,\arg\max_{a\in A}Q_i(s,a))]$. Where $\mathbb{E}[Q(s,\pi'_\phi(s))]$ denotes the value of the action given by policy $\pi'_\phi(s)$, $\arg\max_{a\in A}Q_i(s,a)$ denotes the policy with the objective of maximizing Q_i, σ indicates standard deviation and $\sigma(s,a) \propto \tau \propto \Delta_{sa}$.*

Proof. For the maximum action value in the double value functions, it can be divided into two different cases for discussion:

1) $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) < Q_2(s,a_2)$, which is shown in Fig. 3(a). Since the value function is continuous, there exists an action a_{min} that satisfies $a_1 \leq a_{min} \leq a_2$ and makes $\max_{a\in A}(\min_{i=1,2}Q_i(s,a)) = Q_1(s,a_{min}) = Q_2(s,a_{min})$. Since the value function is an unbiased estimate of the true action value with an estimation error of $[-\tau,\tau]$, which has $\sigma(Q(s,a)) \propto \tau$ and $\sigma(Q(s,a_{min})) \leq \sigma(\max_{a\in A}(\min_{i=1,2}Q_i(s,a)))$, where σ is the standard deviation of multiple independent estimates.

Moreover, in $\max_{a\in A}(\min_{i=1,2}Q_i(s,a))$, the maximal and minimal terms do not contradict each other, and we make a distinction in the subscript. It requires first minimizing the two value functions and then finding the action with maximum value in the resulted function $\min_{i=1,2}Q_i(s,a)$. This is mainly to mitigate the action being incorrectly overestimated and propagated forward.

2) $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) > Q_2(s,a_2)$, which is shown in Fig. 3(b). Since $Q_1(s,a_1) = \max_{a\in A}Q_1(s,a) \geq Q_1(s,a_2)$ and $Q_1(s,a_2) > Q_2(s,a_2)$ are satisfied, then we have $Q_1(s,a_1) \geq Q_1(s,a_2) \geq Q_2(s,a_2) \geq Q_2(s,a_1)$. Based on the comparison of the variance between the two actions a_1 and a_2, $\mathbb{E}[\sigma(s,a_{min})] \leq \min_{i=1,2}\mathbb{E}[\sigma(s,a_i)]$ is obtained.

Therefore, considering the impact of estimation error and overestimation bias, there exists actor training objectives with smaller errors than maximizing a single value function, and we identify $\arg\max_{a\in A}(\min_{i=1,2}Q_i(s,a))$ as one of them. Using action a_{min} for Bellman residual calculation will reduce the effect of estimation error while ensuring the selection of a higher value action. □

To better incorporate the overestimation bias control mechanism, we further explicitly use the standard deviation of the two value estimations as a measure of uncertainty. Finally, the training objective of the actor $\pi_\phi(s)$ is shown in Eq. (12).

$$\mathcal{J}(\pi_\phi)_{EAC} = \mathbb{E}_{s\sim\mathcal{B}}[\min_{i=1,2}Q_i(s,\pi_\phi(s)) - \beta\sqrt{\frac{1}{2}\sum_{i=1,2}(Q_i(s,\pi_\phi(s)) - \bar{Q}(s,\pi_\phi(s)))^2}] \tag{12}$$

where, $\bar{Q}$ is the average of multiple estimations. The first term gives a low bias objective for action policy, and the second term further reduces the accumulation of overestimation bias by minimizing the standard deviation. Under the condition that the experience distribution of replay buffer is constant, a more conservative policy can mitigate the overestimation bias Δ_{sa} generation mentioned. As the training times increase, the value estimation will gradually become accurate, ensuring that the true value of optimal action can be found and propagated forward. The crucial aspect of our method is the ability to prevent excessive forward propagation of action values with large estimation error.

4.4. Explorer for more valuable experience

In this subsection, we discuss how to improve the exploratory ability during the environment interaction phase, and introduce the explorer to select more potentially valuable actions.

Besides training the value function with a constant distribution of experiences replay buffer, how to use the limited number of environmental interactions to gain more valuable experience is another work for the agent. Clearly, in the interaction phase with environment, the agent no longer needs a conservative policy. It should choose the action that is most likely to have potentially high value in the current state. Even many studies have shown that rarely visited actions are more desirable to explore [26,27]. Therefore, we may need a different policy with the optimization process. An inaccurately estimated state reflects insufficient visit to it, which may potentially have a higher true value.

Property 2. *There exists an action selection policy that has a higher potential exploration value than the objective of maximizing a single value function. i.e. There exists a $\pi'_\psi(s)$ satisfies selecting actions with Q values no less than the original policy: $\mathbb{E}[Q(s,\pi'_\psi(s))] = \mathbb{E}[Q(s,\arg\max_{a\in A}Q_1(s,a))] = \mathbb{E}[Q(s,\arg\max_{a\in A}Q_2(s,a))]$ and the action policy $\pi'_\phi(s)$ has smaller variance: $\mathbb{E}[\sigma(s,\pi'_\psi(s)] \geq \max_{i=1,2}\mathbb{E}[\sigma(s,\arg\max_{a\in A}Q_i(s,a))]$, where $\sigma(s,a) \propto \tau \propto \sqrt{3\omega^2/n_{sa}}$, σ indicates standard deviation.*

Proof. For the maximum action value in the double value functions, it can be divided into two different cases for discussion:

1) $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) < Q_2(s,a_2)$, which is shown in Fig. 3(a). There exists an action $a_{max} = \arg\max_{a\in A}(\max_{i=1,2}Q_i(s,a))$ that satisfies $\mathbb{E}[\sigma(s,a_{max})] = \mathbb{E}[\sigma(s,a_1)] = \mathbb{E}[\sigma(s,a_2)]$ and $\mathbb{E}[Q(s,a_{max})] \geq \max_{i=1,2}\mathbb{E}[Q(s,a_i)]$. We can use the max operation to select an action with higher estimation among the two alternative optimal actions to guarantee the most ambient exploration.

2) $Q_1(s,a_1) > Q_2(s,a_1)$ and $Q_1(s,a_2) > Q_2(s,a_2)$, which is shown in Fig. 3(b). Since we already have $Q_1(s,a_1) \geq Q_1(s,a_2) \geq Q_2(s,a_2) \geq Q_2(s,a_1)$, then both $\mathbb{E}[\sigma(s,a_{max})] \geq \max_{i=1,2}\mathbb{E}[\sigma(s,a_i)]$ and $\mathbb{E}[Q(s,a_{max})] \geq \max_{i=1,2}\mathbb{E}[Q(s,a_i)]$ are satisfied. As the estimation error is proportional to the number of training and visits, a_{max} can give a fewer visited and higher estimated action.

Therefore, considering the connection between the number of state visits and the potential exploration value, there exist action selection policies with higher potential exploration value than maximizing a single value function, and we identify $\arg\max_{a\in A}(\max_{i=1,2} Q_i(s,a))$ as one of them. □

In the environmental interaction process, we construct an exploration policy $\pi_\psi(s)$ to perform better action selection instead of the actor, and we name it explorer. The training objective is shown in Eq. (13).

$$\mathcal{J}(\pi_\psi)_{EAC} = \mathbb{E}_{s\sim\mathcal{B}}[\max_{i=1,2} Q_i(s,\pi_\psi(s))] \tag{13}$$

In addition, we do not add a standard deviation term to this objective. Because we only need to ensure the accuracy of the value function corresponding to the optimal policy, and some of the suboptimal states and actions we may not need to visit frequently to compute their values.

4.5. Action mixing mechanism

In this subsection, we discuss the experience distribution bias of the actor-critic method and introduce the action mixing mechanism to mitigate the bias.

In the actor-critic method, both the value network and the policy network are constantly updated and optimized. Random noise needs to be added in the interaction phase to ensure the exploration ability, so the consistency of the experience distribution cannot be guaranteed. However, by controlling the training speed and exploration range, they can ensure that the differences of experience distribution are within a reasonable scope.

When we use two different objectives to train the actor and the explorer, some tricks are also needed to constrain the differences in experience distribution. Specifically, we use action mixing to prevent negative effects of OOD state-action pairs.

Our method can converge to the optimal value function when the training times are sufficient. But considering the approximation error of the networks and the limitation of the sample, the difference in the experience distribution arises mainly for the following two reasons:

1) When the interactions are insufficient, the value estimation for the unvisited state-action pairs is difficult;

2) When training times are insufficient, the policy networks cannot complete the training objectives correctly.

Both of these cases arise mainly in the early stages of the actor-critic methods. As training progresses, both the current actor and explorer gradually converge to the optimal action policy, and the differences in experience distribution gradually decrease. We use action mixing to change the actions selected during environmental interactions, which guarantees that the policy at interaction does not differ too much from the policy at training. The corresponding action selection policy is shown in Eq. (14).

$$a_e = \begin{cases} \pi_\phi(s), & \text{rand}() > p \\ \pi_\psi(s), & \text{otherwise} \end{cases} \tag{14}$$

where rand() denotes the random process obtained by sampling in a uniform distribution $U(0,1)$, p controls the selection proportion of the actor and explorer actions. In practice, we initialize p to a small value of 0.2 and gradually increase it to 1 as the number of training steps t increases.

4.6. EAC method for TD3 and SAC algorithms

In summary, we give all the detailed schemes of the EAC method. Subsequently, we show how the EAC method can be applied to the baseline algorithms.

The EAC method with an actor and an explorer divides the work of the original actor into two parts: value calculation and environment interaction. This method introduces different objectives for more stable training of the value function and better environment exploration, and uses action mixing to mitigate experience distribution bias. It can be applied to actor-critic algorithms, such as DDPG, TD3 and SAC. The main difference with the baseline algorithms is that we need to use actor and explorer networks with different objectives, instead of the original actor network.

Since the TD3 algorithm is widely used, we show how the EAC method can be applied to the TD3 algorithm and obtain the EAC-TD3 algorithm. The EAC-TD3 algorithm does not change the ultimate goal of the value function, but improves the training efficiency by giving a better choice of movements. We give the training objectives for the explorer, actor, and critic of EAC-TD3 as shown in Eq. (15), where we denote $\mathbb{E}_{(s,a,s')\sim\mathcal{B}}$ by the shorthand $\mathbb{E}$.

$$\begin{aligned} \mathcal{J}\left(Q_\theta\right)_{EAC-TD3} &= \mathbb{E}[(r(s,a) + \gamma \min_{i=1,2} \hat{Q}_{\theta_i'}(s', \hat{\pi}_{\phi'}(s') + \epsilon) - Q_\theta(s,a))^2] \\ \mathcal{J}(\pi_\phi)_{EAC-TD3} &= \mathbb{E}[\min_{i=1,2} Q_i(s,\pi_\phi(s)) - \beta\sqrt{\frac{1}{2}\sum_{i=1,2}(Q_i(s,\pi_\phi(s)) - \bar{Q}(s,\pi_\phi(s)))^2}] \\ \mathcal{J}(\pi_\psi)_{EAC-TD3} &= \mathbb{E}[\max_{i=1,2} Q_i(s,\pi_\psi(s))] \end{aligned} \tag{15}$$

The procedure of EAC-TD3 is summarized in Algorithm 1.

Algorithm 1 EAC-TD3.

Initialize critic networks Q_{θ_1}, Q_{θ_2}, actor network π_ϕ, and explorer network π_ψ with random parameters θ_1, θ_2, ϕ, ψ
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with action mixing and exploration noise $a = a_e + \epsilon_1$, $\epsilon_1 \sim \mathcal{N}\left(0, \sigma_1^2\right)$, a_e is obtained by Eq. (14), and observe reward r and next state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  Select action with actor and target noise $a' = \pi_{\phi'}(s') + \epsilon_2$, $\epsilon_2 \sim clip(\mathcal{N}\left(0, \sigma_2^2\right), -c, c)$
  Compute $y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', a')$
  Update critics θ_i by $\arg\min_{\theta_i} \sum (Q_{\theta_i}(s, a) - y)^2 / N$
  if t mod d **then**
    Update actor network π_ϕ by $\mathcal{J}(\pi_\phi)_{EAC-TD3}$ according to Eq. (15) with mini-batch
    Update explorer network π_ψ by $\mathcal{J}(\pi_\psi)_{EAC-TD3}$ according to Eq. (15) with mini-batch
    Update target networks: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$, and $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
  end if
end for

In EAC-TD3 algorithm, we retain the optimization mechanisms such as policy delay and target noise of the baseline algorithm. Our method uses the same framework as the baseline algorithm. However, in the training phase, we add the explorer network with the intention of exploring more valuable actions, and give the actor network a more conservative training objective. In the interaction phase, we introduce a reasonable action policy by action mixing.

In addition, to verify the effect of the EAC method applying to other baseline algorithms, we select the SAC algorithm which is based on the maximum entropy optimization to combine with the EAC method. We give the training objectives for the explorer, actor, and critic of EAC-SAC as shown in Eq. (16), where we denote $\mathbb{E}_{(s,a,s')\sim\mathcal{B}}$ by the shorthand $\mathbb{E}$.

$$\begin{aligned}
\mathcal{J}\left(Q_\theta\right)_{EAC-SAC} &= \mathbb{E}[(\gamma(\min_{i=1,2} \hat{Q}_{\theta_i'}(s', \pi_\phi(s')) - \alpha \log P(\pi_\phi(s'))) + r(s,a) - Q_\theta(s,a))^2] \\
\mathcal{J}\left(\pi_\phi\right)_{EAC-SAC} &= \mathbb{E}[Q_\theta\left(s, \pi_\phi(s)\right) - \alpha \log P(\pi_\phi(s)) - \beta \sqrt{\frac{1}{2}\sum_{i=1,2}(Q_i(s, \pi_\phi(s)) - \bar{Q}(s, \pi_\phi(s)))^2}] \\
\mathcal{J}(\pi_\psi)_{EAC-SAC} &= \mathbb{E}[\max_{i=1,2} Q_i(s, \pi_\psi(s))]
\end{aligned} \tag{16}$$

The EAC-SAC algorithm is obtained by optimizing the action selection policy without affecting the maximum entropy calculation, which is shown in Algorithm 2.

Algorithm 2 EAC-SAC.

Initialize critic networks Q_{θ_1}, Q_{θ_2}, actor network π_ϕ, and explorer network π_ψ with random parameters θ_1, θ_2, ϕ, ψ
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with action mixing and exploration noise $a = a_e$, where a_e is obtained by Eq. (14), and observe reward r and next state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  Select action with actor $a' = \pi_\phi(s')$
  Compute $y = r + \gamma(\min_{i=1,2} Q_{\theta_i'}(s', a') - \alpha \log P(\pi_\phi(s')))$
  Update critics θ_i by $\arg\min_{\theta_i} \sum (Q_{\theta_i}(s, a) - y)^2 / N$
  Update actor network π_ϕ by $\mathcal{J}(\pi_\phi)_{EAC-SAC}$ according to Eq. (16) with mini-batch
  Update explorer network π_ψ by $\mathcal{J}(\pi_\psi)_{EAC-SAC}$ according to Eq. (16) with mini-batch
  Update target networks: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
end for

In Algorithm 2, the SAC algorithm uses a random probability distribution to represent the action policy, replaces the noise based action selection mechanism in TD3, and introduces entropy estimates in each training target, but does not prevent the optimization of action selection policies by the EAC method.

5. Experiments

In this section, we first apply EAC method to TD3 algorithm and extensively evaluate our method on benchmark environments, where we use state-of-the-art methods including TD3, SAC, and PPO [14] as baselines. Secondly, we construct noisy reward and sparse reward tasks to analyze the advantages of the EAC method based on both mitigating the overestimation problem and improving the exploration ability. Then, we conduct a detailed ablation study on EAC to investigate the causes of the performance improvement and analyze the sensitivity of the hyperparameter β. Furthermore, we apply EAC to SAC algorithm for an extensive comparison to further illustrate the compatibility and effectiveness of EAC with other optimization methods.

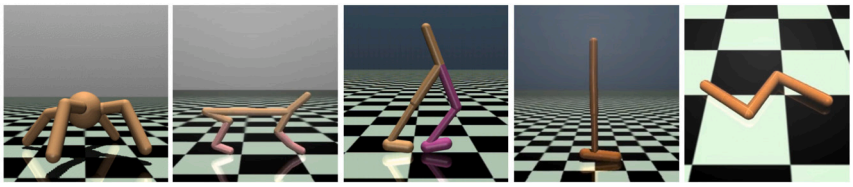


Fig. 4. MuJoCo benchmark environments of Ant-v2, Halfcheetah-v2, Walker2d-v2, Hopper-v2 and Swimmer-v2.

Table 1
Action and state dimensions for environments.

Environment	Action dimensions	State dimensions
Ant-v2	8	111
Walker2d-v2	6	17
HalfCheetah-v2	6	17
Hopper-v2	3	11
Swimmer-v2	2	8

5.1. Experimental tasks and parameter settings

We select five tasks in MuJoCo as the benchmark environment for algorithm evaluation [28,29], which provide simulations of the real world, including friction, gravity, velocity, etc. The agent is a physical model composed of multiple joints, which uses actions to control the movement direction and strength of each joint, observing the current environmental state change and reward feedback to find the optimal action policy that can move forward quickly. The environments are shown in Fig. 4, and the dimensions of action and state are shown in Table 1.

In MuJoCo benchmark environments, the maximum number of interaction steps is 1000 per episode. Some tasks have termination states that bring the episode to an early end, such as Ant, Walker2d and Hopper, where the agent needs to trade off between a stable policy with low reward values and an aggressive policy with higher risks and more rewards.

In the experiments, we mainly used three networks, explorer, actor, and critic. Their training objectives are given in Eq. (15) and Eq. (16). Their network structure is as follows:

- The explorer and actor have the same network structure. They complete the mapping from state space S to action space A through the three-layer fully connected neural network. The dimensions of the input layer and output layer are the same as state space and action space respectively. The dimensions of both the two hidden full connection layers are 256. The activation function is RELU. The learning rate is 0.0003.
- The critic network completes the mapping of state space S to action value Q. Output layer is the one-dimensional cumulative reward estimation and the dimensions of both the two hidden fully connected layers are 256. The activation function is RELU. The learning rate is 0.0003.

We combine the EAC method with TD3 and SAC algorithms to obtain the EAC-TD3 and EAC-SAC algorithms. In order to verify the optimization effect and ensure the generality, the proposed method does not change the hyperparameter settings of the baseline. Since the TD3 algorithm is able to obtain better performance than the original literature after hyperparameter tuning, we use the tuned settings as the benchmark [5]. For the SAC algorithm, we use the same settings as TD3 for the common hyperparameters, and the maximum entropy module is consistent with the literature [12]. The specific parameter settings are shown in Table 2.

In Table 2, the TD3 and SAC specific parameters remain the same as the published parameters of their authors and are not adjusted [5,12]. For the public parameters in "Shard", we refer to the tuning of hyperparameters by the authors of TD3,[1] and also refer to the parameter schemes in the literature [13] and [30] for more stable performance. Our baseline algorithms have better performance than the original settings. We use the same settings for all tasks and do not make special adjustments to a particular algorithm. We are ensuring the fairness and generalization of the experiments.

In addition, for the PPO algorithm, we use three general tricks in the implementation: Normalize advantage value, data processing, and KL divergence constraint to improve the practical performance and stability [31].

5.2. EAC-TD3 performance comparison

In this subsection, we focus on verifying the performance improvement of the EAC method. We compare the EAC-TD3 algorithm with other baseline algorithms on five tasks, and the experimental results are shown in Fig. 5.

In Fig. 5, we use five seeds in each task for three million (3M) steps of interaction and training. "Performance" in the y-axis denotes the sum reward for each episode of cumulative environment rewards for each episode, as $\sum_{k=1}^{n} r_t$. "TotalEnvInteracts" in the

[1] https://github.com/sfujim/TD3.

Table 2
Hyperparameters of algorithms.

Parameter	Value
Shard	
optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers	2
number of hidden units per layer	256
number of samples per mini-batch	256
nonlinearity	ReLu
target smoothing coefficient (τ)	0.05
start steps	$2.5 \cdot 10^4$
TD3	
action noise for interaction (σ_1^2)	0.1
target noise for training (σ_2^2)	0.2
target noise clip (c)	0.5
policy delay (d)	2
SAC	
maximum entropy penalty (α)	0.2
EAC	
standard deviation control (β)	2
action mixing (p)	$0.2 + 0.8 \cdot t/T$

x-axis denotes the number of steps to interact with the environment. We evaluate the performance every 4,000 steps, average the performance over all seeds and use the moving average window of 20 iterations for figures. The solid line and shaded part in each episode reward figure represent the average episode reward sum and its standard deviation for all seeds, respectively. We use the same performance calculations in all experiments.

We can demonstrate that the EAC-TD3 algorithm can effectively improve the training speed and the final reward of the baseline algorithm and outperform all the baseline algorithms. This conclusion is the most evident in Ant and Walker2d, where the two tasks are more complex and the baseline algorithms still have a large potential for improvement. The performance of the existing actor-critic methods in Hopper is close to the optimal result.

Meanwhile, the SAC algorithm performs well in HalfCheetah but struggles to work in Swimmer. This is because the introduction of maximum entropy is difficult to cope with the small gap of different action rewards in Swimmer, and is more suitable to deal with tasks where rewards are easily obtained. It also indicates that the benchmark tasks can detect the performance of the algorithm comprehensively.

In addition, we did not include EAC-SAC in the Fig. 5 comparison because we trained all algorithms using the same common parameters in the environments. We mainly show that EAC-TD3 is able to outperform other algorithms completely. SAC and EAC-SAC algorithms require special tuning in some environments, such as swimmer. Therefore, we use the EAC-SAC algorithm as an additional experiment to demonstrate the generalization of the EAC method and put the results in the summary table.

5.3. Exploration and exploitation improvement

In this subsection, we focus on the two perspectives of exploration and exploitation, and verify the effectiveness of the EAC method in improving exploration and exploitation abilities, respectively.

In order to better highlight the EAC method in mitigating overestimation bias and improving exploration ability, we modify the environment reward function with noisy reward and sparse reward functions [32].

Noisy reward task: The reward function is set as a random reward with mean 0, distribution function is $r(s,a) \sim \mathcal{N}(0,1)$, the number of episode steps is fixed to 1000 with no termination state. The noisy reward will produce larger estimation errors for the value function optimization, as a way to verify the generation of overestimation bias and the mitigation ability of the EAC method.

Sparse reward task: The reward function is based on the center of mass (COM) distance, and the distance reaches m_s will get a reward of 1. The m_s setting is shown in Table 3. The sparse reward task requires more exploratory ability of the agent to obtain rewards and avoid getting into termination states.

5.3.1. Mitigating overestimation bias for exploitation

We verify the overestimation bias problem using the noisy reward tasks, where a noisy reward value would introduce a more estimation bias, while the true value of the average action value would be zero. It has been shown that DDPG is subject to overestimation problem [5] and that TD3 is subject to underestimation problem [11]. Additionally, the introduction of entropy in SAC makes the action value difficult to measure. So we use the EAC method to enhance the DDPG and TD3 algorithms, to test the effectiveness of the EAC method in controlling the estimation bias. The experimental results on the noisy reward task of HalfCheetah are shown in Fig. 6.

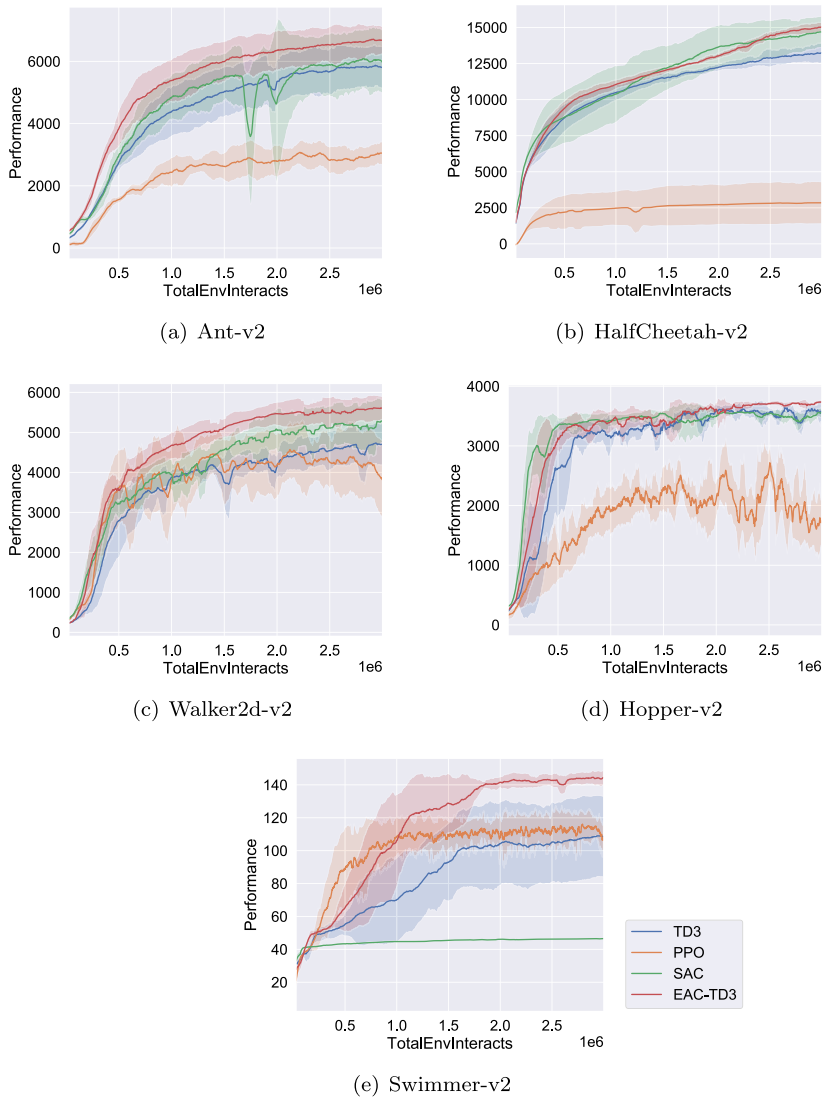


Fig. 5. Average performances comparison on MuJoCo tasks. TotalEnvInteracts denotes the number of steps to interact with the environment, and performance denotes the sum reward for each episode.

Table 3
m_s setting of sparse MuJoCo.

Environment	Centroid distance m_s
Ant-v2	1
Walker2d-v2	1
HalfCheetah-v2	15
Hopper-v2	1

In Fig. 6, "ActionValue" denotes the mean of the Q-value estimates given by the critic network. Affected by estimation error, the DDPG algorithm valuation is greater than 0, resulting in overestimation bias, while the TD3 algorithm suffers from underestimation bias. At the early stage of training, the estimation error is always larger due to the insufficient number of samples. Although the estimation bias is continuously alleviated with sampling and training, it is difficult to converge to the true value 0. The action value estimates of the EAC-DDPG and EAC-TD3 algorithms are closer to the true value. The estimation bias between DDPG and EAC-DDPG action values converges to 9.54 and 6.15, and the estimation bias between TD3 and EAC-TD3 action values converges to -4.89 and -3.92. The overestimation and underestimation problems are mitigated about 36% and 20%, respectively.

Therefore, the conservative action selection policy in the EAC method can mitigate the overestimation or underestimation problems without changing the optimization objective of the value function and is generalized.

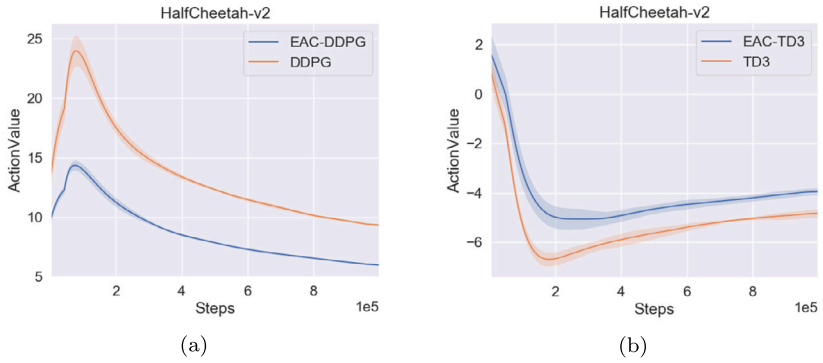


Fig. 6. Average action value estimation under noisy reward tasks.

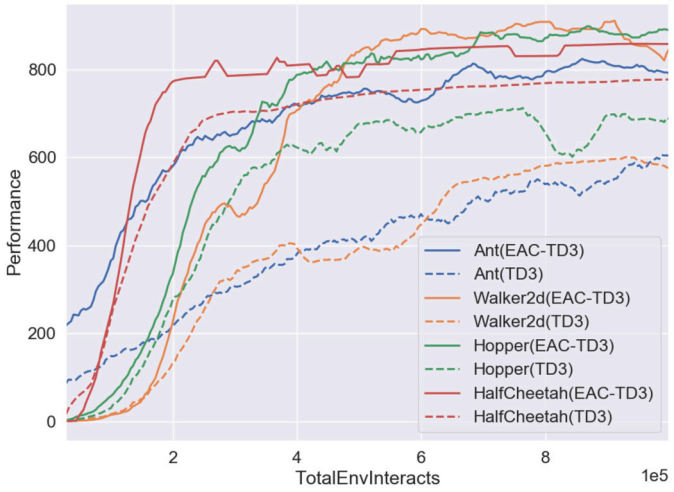


Fig. 7. Performance comparison of EAC-TD3 and TD3 algorithms under sparse reward task.

5.3.2. The effect of exploration enhancement

We verify the effect of exploration ability enhancement by sparse reward tasks. The original intensive rewards include instruction for the agent on staying healthy, controlling cost, and contact force. But sparse reward tasks require the agent with more exploration ability to complete the goal of moving forward. The experimental results using the EAC-TD3 and TD3 algorithms under various sparse reward tasks are shown in Fig. 7.

In Fig. 7, the performance of the EAC-TD3 and TD3 methods are shown by four sets of solid and dashed lines in the same color. The EAC method has faster training efficiency and better final performance on each task. Therefore, the EAC method enhances the exploration ability of agent, which improves sample efficiency and avoids prematurely falling into suboptimal policies.

5.4. Ablation study

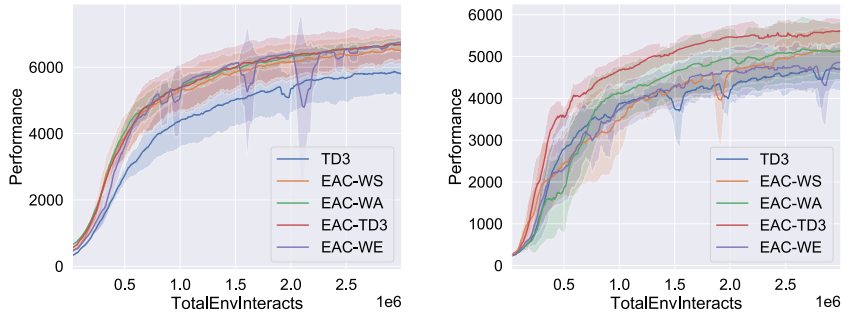
In this subsection, we replace the elements of the EAC method and verify the effectiveness of each optimization on the training performance.

We conduct ablation experiments on two typical tasks: Ant and Walker2d. The improvement of our method can be divided into three parts: the objective change of the actor, the introduction of the explorer network and the action mixing mechanism. We constructed three algorithms to perform the ablation experiments for each part.

EAC-WA: We remove the action mixing mechanism, directly use a conservative policy for Bellman residual, and use an aggressive policy to complete the environmental exploration. This algorithm will verify the effect of potential empirical distribution bias.

EAC-WE: We remove the explorer network and use a conservative actor policy to accomplish both the computation of the Bellman residual and the exploration of the environment. Since there is no explorer network, this algorithm will also remove the action mixing mechanism at the same time, keeping only the objective change of the actor. We can verify how a completely conservative actor contributes to the overall algorithm.

EAC-WS: We remove the minimization standard deviation term which controls the conservativeness of actor and keep only maximizing the minimum of multiple value functions in the objective of actor. It will rarely introduce any hyperparameters. Considering



(a) Ant-v2 (b) Walker2d-v2

Fig. 8. Average performances comparison of several variants of the EAC method.

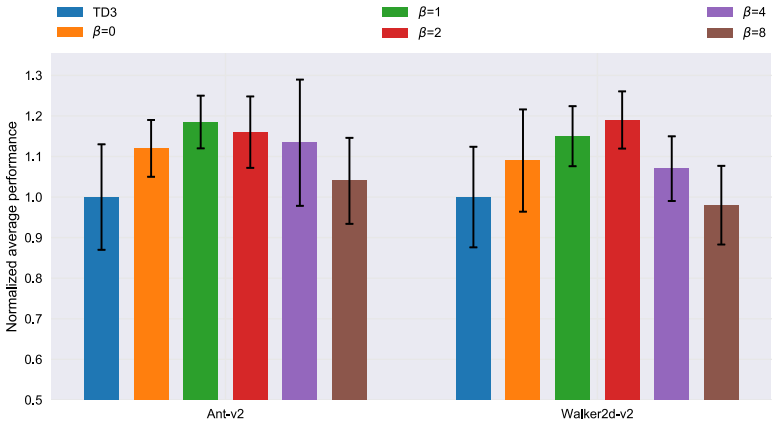


Fig. 9. Normalized average performance in EAC-TD3 with different hyperparameter settings, we set β to $\{0, 1, 2, 4, 8\}$.

the heavy hyperparameter tuning work of deep reinforcement learning algorithms, it may be simplified and practical. This algorithm can verify the effect of the standard deviation term on the performance.

The results of the ablation experiments are shown in Fig. 8, where we compare the performance of several variants of our method with TD3 algorithm.

For the Ant task, all variants of our method have better final performance than TD3 algorithm. They all have a conservative actor policy, only in EAC-WS we weaken the control over estimation error. Meanwhile, the results of EAC-WA and EAC-TD3 are almost identical, and more policy breaks occur in EAC-WE during the training process.

For the Walker2d task, EAC-WE has the least performance improvement as the algorithm with the least difference from TD3. The performance of EAC-WA and EAC-WS is better than TD3 algorithm, but weaker than EAC-TD3. In addition, EAC-WA has the worst performance in the early stage.

Therefore, improving the data efficiency by controlling the estimation error is a crucial factor affecting the performance. Then, the action mixing mechanism can ensure the training efficiency in the presence of large estimation errors. Meanwhile, a better exploration ability can make the training process more stable.

5.5. Hyperparameter sensitivity

Our method regulates the penalty to estimation error by a hyperparameter β, thus taking into account both action value and estimation error in action selection policy. We use multiple β settings on two tasks to verify the variation in performance, which is shown in Fig. 9.

In Fig. 9, we can conclude that the change of β affects the performance of the algorithm. When the β value is small, there exists a more conservative policy of actor that can obtain better training efficiency. When the value is larger, limited by the number of interactions and training, an extremely conservative policy will reduce the training speed. Therefore, an appropriate hyperparameter setting can lead to a better performance, which validates the effectiveness of changing the actor objective. In addition, better results are obtained in Ant with $\beta = 1$ and in Walker2d with $\beta = 2$, because the optimal settings of hyperparameters are also related to the characteristics of the environment in real tasks. But we set $\beta = 2$ in all tasks for fairness, and the other settings are also the same as the baseline.

Table 4
Numerical performance of average return on 3M steps.

		Ant	Walker2d	HalfCheetah	Hopper	Swimmer
PPO	Mean	3074	3847	2845	1818	107
	Std	412	1087	1622	434	11
TD3	Mean	5781	5281	13246	3563	109
	Std	756	598	817	255	27
EAC-TD3	Mean	**6707**	**5606**	**15037**	**3739**	**144**
	Std	**511**	**331**	**406**	**34**	**5**
SAC	Mean	5969	4699	14682	3555	46
	Std	1187	585	1234	72	0.17
EAC-SAC	Mean	**6752**	**5725**	**16078**	**3632**	**47**
	Std	**121**	**155**	**202**	100	0.26

5.6. Extensive experiments

To verify the compatibility of our method with other optimization mechanisms, we apply EAC to the SAC algorithm to obtain the EAC-SAC algorithm. We do not change the optimization mechanism of SAC algorithm, such as introducing maximum entropy and controlling the estimation error in the objective of the actor, which obtains a more conservative policy. We give the performance of EAC-SAC as shown in Table 4.

In Table 4, we indicate the EAC method applied to the baseline algorithm yields better performance in bold, and use the box to mark the highest performance under the current task.

Among them, the EAC-SAC algorithm performs better than the SAC algorithm for each task, and we can conclude that the EAC method can improve the performance of the baseline algorithm, which can be compatible with other optimization mechanisms. In addition, we find that EAC-SAC performs better on Ant, Walker2d and HalfCheetah, and EAC-TD3 performs better on Hopper, and Swimmer, which is caused by the different performance of the SAC and TD3 algorithms.

6. Conclusion and future work

In this paper, we focus on how to optimize the action selection policy to obtain better performance, and do not change the objective of critic.

Firstly, we consider the effects of estimation bias and random noise on off-policy reinforcement learning methods. We confirm the existence of more conservative action policies that can mitigate overestimation bias of the value function, and the existence of exploration policies that can select more exploration-worthy actions.

Then, we propose the Explorer-Actor-Critic (EAC) method, which uses an actor to give more conservative policies and an explorer to select more worthy actions to explore. In addition, we use an action mixing mechanism to reduce the negative effects of experience distribution bias.

In experiments, we apply the EAC method to the TD3 and SAC algorithms, demonstrate the improvement of the EAC method on exploration and exploitation capabilities by noisy reward tasks and sparse reward tasks, and verify its effectiveness through extensive comparison and ablation experiments. Our method not only improves the performance of baseline algorithms, but also can be used on a variety of actor-critic methods.

It can be concluded that the EAC method is able to improve the sample efficiency of the baseline algorithm by changing the action selection policy, optimize the exploration and exploitation capabilities, obtain better performance, and is generalizable.

Future work can be explored from the following aspects:

- Most of the current algorithms evaluate the performance in MuJoCo tasks, and there are already relatively mature ranges of hyperparameter settings and network structures. However, in other tasks, such as real tasks or classical tasks, gradually more complex algorithms lead to a more difficult tuning of the experimental settings. We will also apply these algorithms to a wider variety of tasks in the future.
- Research the relationship between action policy and distribution of experience buffer, and directly correct the negative impact of out-of-distribution samples on value function training through a more accurate distribution calculation.
- Optimize action exploration by constructing auxiliary tasks, and combine environmental characteristics with state values to further improve exploration.
- Focus on the commonality among optimization mechanisms, simplify the model complexity, reduce the difficulty of algorithm reproduction and application, and reduce the cost of hyperparameter tuning.

CRediT authorship contribution statement

Junwei Zhang: Conceptualization, Formal analysis, Methodology, Software, Validation, Writing – original draft. **Shuai Han:** Methodology, Validation, Writing – review & editing. **Xi Xiong:** Software, Validation, Writing – review & editing. **Sheng Zhu:**

Formal analysis, Validation, Writing – review & editing. **Shuai Lü:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

We sincerely thank the anonymous reviewers for their careful work and thoughtful suggestions, which have greatly improved this article. This work was supported by the Natural Science Research Foundation of Jilin Province of China under Grant Nos. 20220101106JC and YDZJ202201ZYTS423, the National Natural Science Foundation of China under Grant No. 61300049, the Fundamental Research Funds for the Central Universities (Jilin University) under Grant No. 93K172022K10, the Fundamental Research Funds for the Central Universities (Northeast Normal University) under Grant No. 2412022QD040, and the National Key R&D Program of China under Grant No. 2017YFB1003103.

References

[1] M. Hessel, J. Modayil, H. Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M.G. Azar, D. Silver, Rainbow: combining improvements in deep reinforcement learning, in: The 32nd AAAI Conference on Artificial Intelligence (AAAI 2018), Louisiana, USA, 2018, pp. 3215–3222.
[2] W. Dabney, G. Ostrovski, D. Silver, R. Munos, Implicit quantile networks for distributional reinforcement learning, in: The 35th International Conference on Machine Learning (ICML 2018), Stockholm, Sweden, 2018, pp. 1104–1113.
[3] S. Han, Y. Sung, Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning, in: The 36th International Conference on Machine Learning (ICML 2019), California, USA, 2019, pp. 2586–2595.
[4] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: The 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2016.
[5] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: The 35th International Conference on Machine Learning (ICML 2018), Stockholm, Sweden, 2018, pp. 1582–1591.
[6] H. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: The 30th AAAI Conference on Artificial Intelligence (AAAI 2016), Arizona, USA, 2016, pp. 2094–2100.
[7] K. Ciosek, Q. Vuong, R. Loftin, K. Hofmann, Better exploration with optimistic actor critic, in: The 32nd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 2019, pp. 1785–1796.
[8] X. Chen, C. Wang, Z. Zhou, K.W. Ross, Randomized ensembled double Q-learning: learning fast without a model, in: The 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, Austria, 2021.
[9] Z. Zhang, Z. Pan, M.J. Kochenderfer, Weighted double Q-learning, in: The 26th International Joint Conference on Artificial Intelligence (IJCAI 2017), Melbourne, Australia, 2017, pp. 3455–3461.
[10] O. Anschel, N. Baram, N. Shimkin, Averaged-DQN: variance reduction and stabilization for deep reinforcement learning, in: The 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 2017, pp. 176–185.
[11] Q. Lan, Y. Pan, A. Fyshe, M. White, Maxmin Q-learning: controlling the estimation bias of Q-learning, in: The 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 2020.
[12] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: The 35th International Conference on Machine Learning (ICML 2018), Stockholm, Sweden, 2018, pp. 1856–1865.
[13] L. Pan, Q. Cai, L. Huang, Softmax deep double deterministic policy gradients, in: The 33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Virtual Event, 2020.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347, 2017.
[15] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: The 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 2017, pp. 2778–2787.
[16] Y. Burda, H. Edwards, D. Pathak, A.J. Storkey, T. Darrell, A.A. Efros, Large-scale study of curiosity-driven learning, in: The 7th International Conference on Learning Representations (ICLR 2019), New Orleans, USA, 2019.
[17] G. Ostrovski, M.G. Bellemare, A. Oord, R. Munos, Count-based exploration with neural density models, in: The 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 2017, pp. 2721–2730.
[18] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F.D. Turck, P. Abbeel, #exploration: a study of count-based exploration for deep reinforcement learning, in: The 30th Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, USA, 2017, pp. 2753–2762.
[19] S. Fujimoto, D. Meger, D. Precup, Off-policy deep reinforcement learning without exploration, in: The 36th International Conference on Machine Learning (ICML 2019), Long Beach, USA, 2019, pp. 2052–2062.
[20] A. Kumar, J. Fu, M. Soh, G. Tucker, S. Levine, Stabilizing off-policy Q-learning via bootstrapping error reduction, in: The 30th Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 2019, pp. 11761–11771.
[21] C. Zhang, S.R. Kuppannagari, V.K. Prasanna, BRAC+: improved behavior regularized actor critic for offline reinforcement learning, in: Asian Conference on Machine Learning (ACML 2021), Virtual Event, 2021, pp. 204–219.
[22] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative Q-learning for offline reinforcement learning, in: The 33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Virtual Event, 2020.
[23] S. Han, W. Zhou, S. Lü, S. Zhu, X. Gong, Entropy regularization methods for parameter space exploration, Inf. Sci. 622 (2023) 476–489.
[24] S. Han, W. Zhou, S. Lü, J. Yu, Regularly updated deterministic policy gradient algorithm, Knowl.-Based Syst. 214 (2021) 106736.
[25] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, second edition, MIT Press, 2018.

[26] Y. Burda, H. Edwards, A.J. Storkey, O. Klimov, Exploration by random network distillation, in: The 7th International Conference on Learning Representations (ICLR 2019), New Orleans, USA, 2019.
[27] M.G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, in: The 29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 2016, pp. 1471–1479.
[28] D. Zhang, T. Zhang, S. Jia, B. Xu, Multi-sacle dynamic coding improved spiking actor network for reinforcement learning, in: The 34th AAAI Conference on Artificial Intelligence (AAAI 2022), Virtual Event, 2022, pp. 59–67.
[29] E. Nikishin, M. Schwarzer, P. D'Oro, P. Bacon, A.C. Courville, The primacy bias in deep reinforcement learning, in: The 39th International Conference on Machine Learning (ICML 2022), Baltimore, Maryland, USA, 2022, pp. 16828–16847.
[30] X. Gong, S. Lü, J. Yu, S. Zhu, Z. Li, Adaptive estimation Q-learning with uncertainty and familiarity, in: The 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023), Macao, SAR, China, 2023, pp. 3750–3758.
[31] J. Zhang, Z. Zhang, S. Han, S. Lü, Proximal policy optimization via enhanced exploration efficiency, Inf. Sci. 609 (2022) 750–765.
[32] D. Zha, W. Ma, L. Yuan, X. Hu, J. Liu, Rank the episodes: a simple approach for exploration in procedurally-generated environments, in: The 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, Austria, 2021, pp. 387–395.