Information Sciences 647 (2023) 119494

Contents lists available at ScienceDirect

Information Sciences

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

Multi-actor mechanism for actor-critic reinforcement learning

Lin Li, Yuze Li, Wei Wei [*], Yujia Zhang, Jiye Liang

Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi, China

A R T I C L E   I N F O

Keywords:
Reinforcement learning
Multi-actor
Actor-critic
Value estimation

A B S T R A C T

Value estimation is a critical problem in Value-Based reinforcement learning. Most related studies focus on using multi-critic to reduce estimation bias and seldom consider the multi-actor impact on value estimation. This paper proposes a multi-actor mechanism (MAM) for Actor-Critic reinforcement learning that can provide multiple behavior choices in the same state, resulting in diverse Q-values that provide richer information and enhance exploration capability. MAM contains two technologies. One is obsolescence technology, which quickly generates high-quality experience to help the agent find the optimal policy. The other is Q-value weighting technology, which leverages multiple Q-values to achieve a more accurate value estimation. The proposed mechanism, MAM, is general and can be applied to any Actor-Critic reinforcement learning algorithm. Specifically, we embed MAM into DDPG and TD3 and demonstrate that MAM can mitigate estimation bias, enhance exploration, and yield state-of-the-art results in various MuJoCo tasks, including the challenging Humanoid-v2 and Walker2d-v2 benchmarks.

1. Introduction

Reinforcement Learning (RL), a paradigm in machine learning, differs from traditional supervised and unsupervised learning. RL aims to obtain the optimal policy through trial and error [1]. Combining Deep Learning (DL), Deep Reinforcement Learning (DRL) has recently performed well in many complex tasks [2], such as playing games [3–5], robot control [6,7], financial trading [8], recommendation [9,10], and so on. However, several challenging issues still prevent DRL from being applied to a broader range of tasks. The value function's estimation bias is critical in these challenging issues.

Many popular RL algorithms, such as Deep Q-Network (DQN) [5,11], suffer from overestimation problems, usually caused by the maximization operation, insufficiently flexible approximation [12], or approximation noise [13]. The essence of temporal difference learning [1] further exaggerates the estimation bias because using the estimation of a subsequent state to update an estimate of the value function. To solve this problem, Fox et al. [14] and Lee et al. [15] try to add a penalty or correct the policy. Nachum et al. [16] apply the smoothed value functions to lower bias. Wang et al. [17] use the advantage function to relieve overestimation. These improvements above focus on the single critic. Some approaches use multi-critic to reduce the overestimation bias. Double Q-learning [13] and Double DQN [18] alleviate the overestimation problem by using two critics. Anschel et al. [19] try to minimize estimation error by averaging multiple Q-values directly. From the literature mentioned above, using multi-critic can effectively alleviate the overestimation problem. The above approaches focus on the discrete action settings. At the same time, the Deep Deterministic Policy

[*] Corresponding author.
E-mail addresses: lilynn1116@sxu.edu.cn (L. Li), 202022407033@sxu.edu.cn (Y. Li), weiwei@sxu.edu.cn (W. Wei), 342564535@qq.com (Y. Zhang), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.ins.2023.119494
Received 10 December 2022; Received in revised form 30 May 2023; Accepted 7 August 2023

Available online 10 August 2023

Gradient algorithm (DDPG) [20], an Actor-Critic method to solve continuous action tasks, also suffers from overestimation in the continuous control settings.

Twin Delayed Deep Deterministic policy gradient (TD3) [21] is a widely-used RL algorithm for continuous control tasks. By taking the minimization operation of two critics, TD3 relieves the overestimation bias problem in DDPG. However, TD3 can cause an underestimation bias at each updating iteration due to the minimization operation. Although this bias is not explicitly propagated during the updating process, the inaccurate estimation still negatively affects the performance of DL algorithms. Thus, some researchers study how to balance the overestimation bias and the underestimation bias [22,23]. Others use the softmax operation in updating value [24], or quasi-median operation [25], or proposes triplet-average deep deterministic policy gradient (TADD) [22] to reduce the estimation bias. Existing works primarily focus on the improvement of critics, with little attention to the effect of actors on value estimation. Lyu et al. [26] propose Double Actors Regularized Critics (DARC) to enhance exploration ability by selecting the action with the larger Q-value from the two actors and relieving the estimation bias using the regularization method. However, the impact of inaccurate value estimation may be further aggravated by action selection based on maximum Q-value. Besides, collaborative evolutionary reinforcement learning (CERL) [27] uses a collaborative approach of learners to improve the model's performance. However, the CERL algorithm uses a portfolio of TD3 learners, which also suffers from server underestimation bias and neglects the impact of learner numbers on value estimation.

To explore how the multi-actor can effectively enhance the exploration ability of the agent and achieve accurate value function estimation, we propose a multi-actor mechanism (MAM) oriented to the Actor-Critic framework in RL. Specifically, the MAM contains two technologies: obsolescence technology (OT) and Q-value weighting technology (WT). In obsolescence technology, well-performed actors can generate more high-quality experience to help agents obtain the optimal policy faster. The weighting technique can mitigate underestimation by weighting the Q-values of actions generated by a different actor in the same state. We first propose a Multi-Actor Deep Deterministic Policy Gradient (MA-DDPG) based on DDPG, showing the effectiveness of the obsolescence technology on artificial experiments. Then, we offer a Multi-Actor Twin Delayed Deep Deterministic policy gradient (MA-TD3) based on TD3 and demonstrate that MA-TD3 can achieve more accurate and stable value estimation. And 8 MuJoCo [28] experiments show that our method outperforms the existing representative methods.

The main contributions are summarized as follows:

- We propose a multi-actor mechanism (MAM) that can be applied to any Actor-Critic algorithm containing two technologies. One is an obsolescence technology that enhances the exploration ability and allows the agent to find the optimal strategy faster. The other is a Q-value weighting technology based on multi-actor, which can relieve both estimation bias and variance.
- We prove that the algorithms embedded in the MAM can achieve a more accurate and stable value estimation.
- We embed the MAM into the typical Actor-Critic algorithms, DDPG and TD3, to propose MA-DDPG and MA-TD3. Numerous experiments have shown that the algorithms embedded with MAM can improve exploration ability, relieve estimate bias, and enhance the algorithm's performance.

2. Preliminaries

RL considers the paradigm of an agent interacting with its environment to learn return-maximizing behavior. RL can be formulated by a Markov Decision Process (MDP), defined by a tuple $M = (S, A, p, p_0, R, \gamma)$ consisting of a state space S, an action space A, a transition kernel p, an initial state distribution p_0, a reward function R, and a discount factor $\gamma \in [0, 1]$. The transition kernel is the probability distribution of transferring to all possible states after executing action A in state S. The cumulative reward is defined as $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$.

In a Value-Based RL algorithm, the state-action function (or Q-function) is defined as $Q(s, a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a]$. The state-action function is used to evaluate the policy, which means that precise estimated value plays an essential role in learning a better policy. In most cases, the transition probability function is unknown, and thus Q-function can be approximated using the following equation:

$$Q(s, a) = r + \gamma E_{s' \sim S, a' \sim \pi}[Q(s', a')]. \tag{1}$$

2.1. Deep deterministic policy gradient

The DDPG algorithm, which is designed by combining Deep Neural Network (DNN) and Deterministic Policy Gradient (DPG) [29], uses the deterministic policy instead of the stochastic policy. DDPG is a more efficient method to estimate value functions than stochastic policy. Besides, it trains a deterministic policy actor by using a learned value estimator critic, which means that deterministic policy specifies one action with the guidance of a single critic.

Let θ, θ', ϕ and ϕ' denote the parameters of critic Q, target critic Q', actor π, and target actor π', respectively. The update equation of the critic is as follows:

$$L(\theta) = E_{(s,a,r,s') \sim B}[(y - Q(s, a; \theta))^2], \tag{2}$$

where $y = r + \gamma Q'(s', \pi'(s'; \phi'); \theta')$ is the target value based on the target critic network, and B is the replay buffer that stores the past transitions. The chain rule of gradient propagation to update the actor is:

$$\nabla_{\phi} J(\phi) = E_{s\sim p_\pi}[\nabla_a Q(s,a;\theta)|_{a=\pi(s;\phi)} \nabla_{\phi}\pi(s;\phi)], \tag{3}$$

where p_π is a transition probability function based on π. Besides, DDPG uses a soft target update for better stability of the learning process:

$$\phi' = \tau\phi + (1-\tau)\phi', \theta' = \tau\theta + (1-\tau)\theta', \tag{4}$$

where τ is a small enough constant that can adjust the update speed.

Although DDPG performs well on some continuous control tasks, it still cannot solve the overestimation problem due to the gradient ascent [21,22,25].

2.2. Twin delayed deep deterministic policy gradient

The TD3 is a popular RL algorithm that uses the Actor-Critic framework to learn a deterministic policy. It overcomes overestimation in the DDPG by applying two critics for value estimation and taking the minimum between two estimates for target updating. Similar to the DDPG, the TD3 still utilizes the same loss as Equation (2) but $y = r + \gamma min_{i=1,2} Q'_i(s', \pi'(s';\phi');\theta'_i)$, where Q'_1, Q'_2 represent two target critics corresponding to a pair of independent critics Q_1 and Q_2. The update function of the actor in the TD3 is also the same as the one in DDPG, but it chooses a fixed one of the two critics to guide the actor's update.

The TD3 has two additional tricks to improve its performance. One is delaying policy updates, which makes the algorithm more stable. Another is target policy smoothing regularization, which adds noise to the target policy and effectively relieves over-fitting. Those trick makes the TD3 alleviate the overestimation problem and outperforms the DDPG. However, Wu et al. [22] and Wei et al. [25] propose that TD3 still has an underestimation problem even when the approximations are unbiased due to the minimization operation.

3. Multi-actor mechanism

This section introduces MAM and its two core technologies, obsolescence technology and Q-value weighting technology. We show that obsolescence technology enhances the ability of exploration by allowing well-performing actors to generate more experience to enrich the experience pool. And we theoretically prove that the Q-value weighting technology based on multi-actor can alleviate estimation bias and variance.

In MAM, there are K independent actors $\pi_1, \ldots, \pi_K$, and the number of critics varies in different algorithms. When MAM is embedded in a single critic algorithm DDPG, there are K critics $\theta_1, \ldots, \theta_K$. The update function of critics is as follows:

$$L(\theta_k) = E_{(s,a,r,s')\sim B}[(y - Q(s,a;\theta_k))^2], \tag{5}$$

where $y = r + \gamma Q'_k(s', \pi'_k(s'))$, and Q'_k represents target critic corresponding to critic Q_k. The update function of actor π_k is as follows:

$$\nabla_{\phi_k} J(\phi_k) = E_{s\sim p_\pi}[\nabla_a Q(s,a;\theta_k)|_{a=\pi_k(s;\phi_k)} \nabla_{\phi_k}\pi_k(s;\phi_k)]. \tag{6}$$

When MAM is embedded in multi-critic algorithm, there are $n * K$ critics, $\theta_{11}, \theta_{12}, \ldots, \theta_{1n}, \ldots, \theta_{K1}, \theta_{K2}, \ldots, \theta_{Kn}$. The update function of the critic is the same as Equation (5), but the target value y depends on the algorithm itself. For example, in the TD3, $n = 2$, $y_k = r + \gamma min_{i=1,2} Q'_{ki}(s', \pi'(s'))$ and Q'_{k1}, and Q'_{k2} represent two target critics corresponding to a pair of independent critics Q_{k1} and Q_{k2}. The actor π_k is optimized with the estimated value of critic Q_{k1}:

$$\nabla_{\phi_k} J(\phi_k) = E_{s\sim p_\pi}[\nabla_a Q(s,a;\theta_{k1})|_{a=\pi_k(s;\phi_k)} \nabla_{\phi_k}\pi_k(s;\phi_k)]. \tag{7}$$

3.1. Obsolescence technology and its advantage

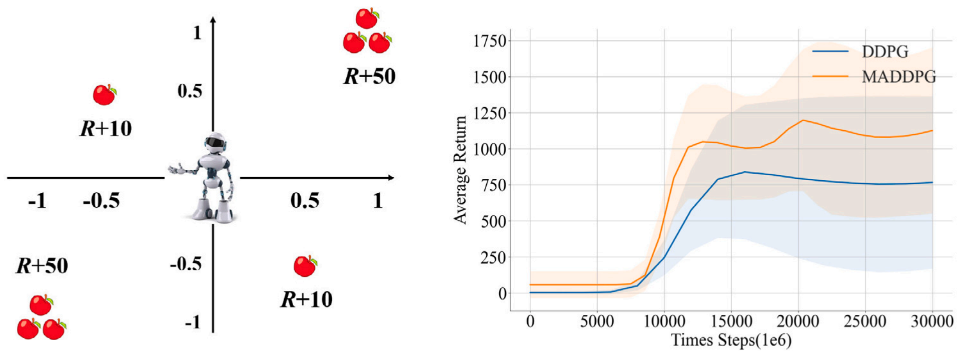
3.1.1. Obsolescence technology

Then, we design an obsolescence technology for better actors to generate more experience in the early training stage, enhancing the exploration ability and can help the agent find optimal policy faster. The essence of the obsolescence technology is to make the good actors generate more experience and eventually eliminate the poor actors. Specifically, we evaluate multiple actors based on average cumulative rewards and select well-performed actors to generate more experience. Only the best actor is finally retained as the policy is continuously updated. So we refer to the above process as the obsolescence technology.

Since multi-actor cannot interact with one environment, we copy M $(M > K)$ environments for multi-actor to generate experience. Meanwhile, we test the performance of K actors every A steps to determine which actor can interact with more environments to obtain a more high-quality experience. We don't eliminate actors that perform unsatisfactorily at the beginning but change the weighting of different actors to pick out the good actors with a high environment interaction probability. The probability, P_k, is defined as:

$$P_k = \frac{R_k}{\sum_{j=1}^{K} R_j}, \tag{8}$$

where $R_1, \ldots, R_K$ are the accumulated reward of K actors in the testing stage. We add up the rewards that agent obtains in one episode as cumulative rewards. For example, in this paper, we test each actor 20 times in the test environment and average the



(a) AppleCollect (b) Comparison results of MA-DDPG and DDPG

Fig. 1. Experimental verification of multi-actor enhance exploration.

Table 1
Hyperparameters comparison of MA-DDPG and DDPG.

Hyperparameter	DDPG	MA-DDPG
Actor network	(256,256)	(256,256)
Critic network	(256,256)	(256,256)
Explore noise	0.1	0.1
Batch size	100	100
Discount	0.99	0.99
Actor learning rate	0.0003	0.0003
Critic learning rate	0.0003	0.0003
Optimizer	Adam	Adam
Target network update rate	0.005	0.005
Start time step	10,000	10,000
Actor number K	None	3
Number of environment M	None	5
Elimination step C	None	24,000
Competition frequency A	None	1,000

20 cumulative rewards as R_k. In each environment, the k-th actor gets an opportunity to interact with the environment based on P_k. In the case of positive rewards the equation can be used to calculate P_k directly. In the case of negative rewards, we take an reverse operation while using the equation. Specifically, after calculating P_k, we sort P_k sequence by probability and then reverse the probability sequence. This operation still ensures that actors with smaller cumulative reward values have less probability of creating experiences. In comparison, actor with larger cumulative reward values are likelier to develop experiences. The final number of interactions with the environment of the k-th actor is an integer, I_k $(0 \leq I_k \leq M)$. In this paper, the number of environment is M. Each actor interacts with the environment in a probability of P_k. And the final number of interactions with the environment of the k-th actor is defined as an integer I_k $(0 \leq I_k \leq M)$. The P_k is uniform in the initial state.

To save computing resources and improve the stability of our algorithm, we define the elimination step as C, which means that the poorly performed actors will not be permitted to participate in the training after the C steps. We choose the best actor based on the average cumulated reward obtained by the actor interacting with the environment.

Compared with a single actor, multi-actor are conducive to early exploration, which makes agents explore high-value areas faster with the same number of steps. Furthermore, our obsolescence technology guarantees that good actors can produce more experience in the early training stage, facilitating the actor to find the optimal policy quickly.

3.1.2. Advantage of obsolescence technology

To verify the proposed MAM, we present the MA-DDPG algorithm and design a artificial two-dimensional, continuous state and action task AppleCollect to test its performance. Fig. 1 (a) shows the artificial environment. The experimental setting is a square area with a side length of 2. The agent starts from the region's center, $x, y = (0, 0)$, with the maximum action range of a single movement of 0.2. There are four locations with apples in the area, two of which are rewarded with 10 and the rest with 50. The positions of the apple area with the ample reward are (-0.8, -0.8) and (0.8, 0.8). The places with a small reward are (0.5, -0.5) and (-0.5, 0.5). We set the maximum number of steps in a single episode as 100. A reward of -0.01 is assigned to the agent after each step. In the MA-DDPG, the number of actors K is 3, indicating that three actors involved in the interaction with the environment. The maximum number of environments M is 5, indicating that the actor can interact with five environments. And the elimination step C is 15,000, which means poorly performed actors will be eliminated after 15,000 steps. See Table 1.

The performance comparison of the MA-DDPG and the DDPG in the AppleCollect is shown in Fig. 1 (b). The darker lines represent the average return of 10 trials, and the shaded area represents the 1 standard deviation. The results indicate that the multi-actor-based

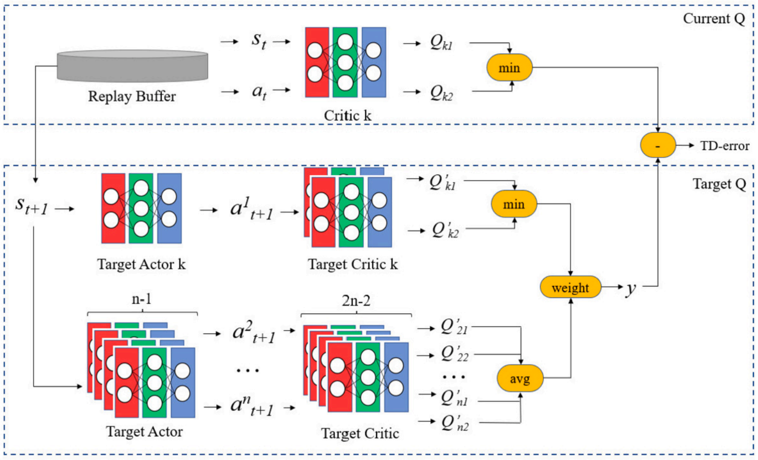


Fig. 2. Visualization of Q-value weighting technology based on MAM, where “min” means a minimum of two values, “avg” means an average of all values, and “weight” means a weight of two values.

agent exhibits higher exploration under the competitive approach, finds regions with higher reward values faster, and ultimately outperforms the algorithms using a single actor in terms of performance.

3.2. *Q-value weighting technology and its advantage*

3.2.1. *Q-value weighting technology*

The accurate value estimation directly affects the algorithm’s performance [21]. To obtain a more accurate value estimation, we propose Q-value weighting technology in MAM. The Q-weighting technique is a weighting idea that uses multiple choices given by a multi-actor to generate diverse Q-value. Then it calculates the target Q-value to relief the estimation bias by weighting the diverse Q-value. Further, we embed the mechanism into the classical Actor-Critic algorithm TD3 and present the Multi-actor TD3 (MA-TD3), which means that MA-TD3 contains both obsolescence technology and Q-value weighting technology. The process of the weighting process and calculating the TD error of MA-TD3 is shown in Fig. 2.

The underestimation problem impairs TD3’s performance to a large extent due to the minimization operation. When calculating the target value, we average the Q-values guided by a multi-actor and combine the averaging operation with the minimization operation, taking a trade-off of overestimation and underestimation. In the MA-TD3, the critic update is based on Equation (5), but

$$y = r + \gamma((1-\eta)\min_{i=1,2} Q_{ki}(s',\pi_k(s')) + \frac{\eta}{2k-2}(\sum_{j=0,\neq k}^{K}(Q_{ji}(s',\pi_j(s'))), \eta \in (0,1). \tag{9}$$

3.2.2. *Advantage of Q-value weighting technology*

In this section, our proposed technique is analyzed from theoretical and experimental aspects. In Theorem 1, we analyze the expectation of the bias and verify that the estimation bias of our methods MA-TD3 $E(B_{ki})$ are upper bounded by $\lambda - \frac{\mu(1-\eta)}{3}$. For details, see below.

Theorem 1. *Let $Q^*(s,a)$ be the true state-action value and suppose that there are K estimation values $\hat{Q}_k(s,a)$. Denote the estimation bias (upper bound) $B_{ki} = \hat{Q}_{ki}(s,a) - Q^*(s,a)$ are independently identical distribution in uniform distribution $U(\lambda-\mu, \lambda+\mu)$, where $0 < \lambda << \mu$. Then,*

$$E(B_{ki})^{MA-TD3} = \lambda - \frac{1}{3}\mu(1-\eta). \tag{10}$$

Proof. From [25], we have the cumulative distribution function and the probability density function of $\min_{i=1,2} B_{ki}$:

$$F(X) = \begin{cases} 0 & x < \lambda-\mu \\ 1-(1-\frac{x-\lambda+\mu}{2\mu})^2 & \lambda-\mu \le x < \lambda+\mu \\ 1 & x \ge \lambda+\mu, \end{cases}$$

$$f(x) = \begin{cases} \frac{1}{\mu}(1-\frac{x-\lambda+\mu}{2\mu}) & \lambda-\mu \le x < \lambda+\mu \\ 0 & else. \end{cases}$$

Then,

Algorithm 1 MA-TD3

```
1: Initialize $2K$ critic networks $Q_{11}, Q_{12}, \cdots, Q_{k1}, Q_{k2}$ and $K$ actor networks $\pi_1, \pi_2, \cdots, \pi_k$ with random parameters $\theta_{11}, \theta_{12}, \cdots, \theta_{k1}, \theta_{k2}, \phi_1, \phi_2, \cdots, \phi_k$
2: Initialize target networks $\theta'_{ki} \leftarrow \theta_{ki}, \phi'_k \leftarrow \phi_k$ and replay buffer $\mathcal{B}$
3: Initialize $I_k$ with average probability of environment interaction $P_k$
4: for $t = 1$ to $T$ do
5:     for each actor $k$ do
6:         for $i_k = 0$ to $I_k$ do
7:             Select action $a_{i_k} \sim \pi_k(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
8:             Observe reward $r$ and new state $s'$
9:             Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$
10:        end for
11:        Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
12:        $a'_k \leftarrow \pi_k(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$ and calculate $y$ in Equation (9)
13:        Update critics $\theta_{ki} \leftarrow arg\min_{\theta_{ki}} N^{-1} \sum (y - Q_{\theta_{ki}}(s, a))^2$
14:    end for
15:    if $t$ mod $A == 0$ and $t < C$ then
16:        Calculate $I_k$ and $P_k$ with Function (8)
17:    end if
18:    if $t == C$ then
19:        Select the best actor $e$ based on the average cumulated reward, and let $k = e, I_k = 1$
20:    end if
21:    if $t$ mod $d == 0$ then
22:        Update $\phi_k$ by Equation (7)
23:        Update target networks by Equation (4)
24:    end if
25: end for
```

$$E(\min_{i=1,2} B_{ki}) = \int_{\lambda-\mu}^{\lambda+\mu} x f(x) dx = \int_{\lambda-\mu}^{\lambda+\mu} \left(\frac{x}{\mu} - \frac{x^2 - \lambda x + \mu x}{2\mu^2}\right) dx = \lambda - \frac{1}{3}\mu. \tag{11}$$

$$E\left(\sum_{j=0, j\neq k}^{K} B_{ji}\right) = (2K-2)\lambda. \tag{12}$$

Combine Equation (11) and Equation (12):

$$\begin{aligned} E(B_{ki}) &= (1-\eta)(\lambda - \frac{1}{3}\mu) + \frac{\eta}{2K-2}(2K-2)\lambda \\ &= \lambda - \frac{1}{3}\mu(1-\eta). \quad \square \end{aligned} \tag{13}$$

Denote the bias's expectations (upper bound) of the MA-TD3 and the TD3 as $E(B_{ki})^{MA-TD3}$ and $E(B_{ki})^{TD3}$. According to Equation (11) and Equation (13), we have:

$$E(B_{ki})^{TD3} = \lambda - \frac{1}{3}\mu, E(B_{ki})^{MA-TD3} = \lambda - \frac{1}{3}\mu(1-\eta).$$

Further, we have:

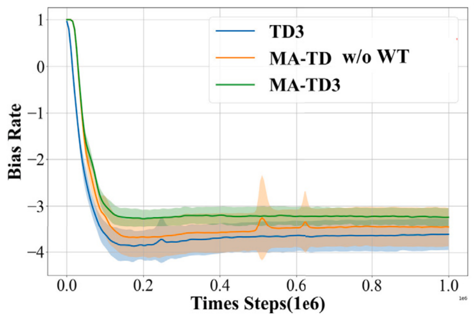
$$\begin{aligned} E(B_{ki})^{TD3} - E(B_{ki})^{MA-TD3} &= \lambda - \frac{1}{3}\mu - \lambda + \frac{1}{3}\mu(1-\eta) \\ &= -\frac{1}{3}\mu\eta < 0, \eta \in (0,1). \end{aligned} \tag{14}$$

According to Equation (14), the bias's expectation (upper bound) of the MA-TD3 is lower than the bias's expectation (upper bound) of the TD3, which indicates that the MA-TD3 relieves the underestimation problem suffered by the TD3.

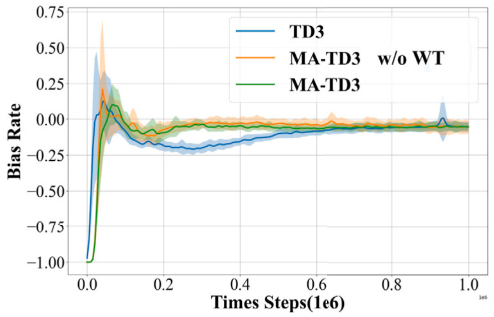
Moreover, we analyze the variance of the MA-TD3 in the different number of actors by Theorem 2, showing the relationship between value estimation stability and the number of actors.

Theorem 2. *Let $Var(B_{ki})_K^{MA-TD3}$ denote the variance of MA-TD3 with K actors. Then:*

$$Var(B_{ki})_m^{MA-TD3} - Var(B_{ki})_n^{MA-TD3} > 0, (m < n).$$



(a) Reacher-v2



(b) Ant-v2

Fig. 3. Estimate bias comparison in two continuous control tasks, where w/o WT represents without weighting technology.

Proof.

$$
\begin{aligned}
Var(B_{ki})_K^{MA-TD3} &= (1-\eta)^2 Var(min_{i=1,2} B_i) + \frac{\eta^2}{(2K-2)^2} Var(\sum_{j=0,j\neq k}^{K} B_{ji}) \\
&= (1-\eta)^2 Var(min_{i=1,2} B_i) + \frac{\eta^2}{(2K-2)^2} \frac{(2K-2)(2\mu)^2}{12} \\
&= (1-\eta)^2 Var(min_{i=1,2} B_i) + \frac{\eta^2\mu^2}{6K-6}.
\end{aligned}
\tag{15}
$$

Thus, we have:

$$
\begin{aligned}
Var(B_{ki})_m^{MA-TD3} - Var(B)_n^{MA-TD3} &= \frac{\eta^2\mu^2}{6m-6} - \frac{\eta^2\mu^2}{6n-6} \\
&= \eta^2\mu^2 \frac{6n-6m}{(6n-6)(6m-6)} > 0. \quad \square
\end{aligned}
\tag{16}
$$

From the above theorems, weighting the Q-values based on multi-actor can reduce the estimation bias and variance when calculating the target Q-values. Furthermore, we demonstrate the improvement of accurate value estimation in the MuJoCo environment. In each MuJoCo continuous control task, we evaluate estimation bias, calculate the difference between the true Q-values and the estimated Q-values of the current critic every 5,000 steps, and use the estimation bias rate as a criterion to indicate the magnitude of the estimation bias. Fig. 3 shows the experimental results, where the darker lines represent the average bias rate of 10 random seeds, and the shaded area represents the one standard deviation. The bias rate is calculated by the function below:

$$
Rate = \frac{\hat{Q}(s,a) - Q^*(s,a)}{Q^*(s,a)}, \tag{17}
$$

where the $Q^*(s,a)$ is the true Q-value estimated by rolling out the current policy 20 times in the initial environment and recording the average discounted reward.

To distinguish the effect of obsolescence technology and Q-value weighting technology on value estimation, we conduct the ablation study on two typical MuJoCo environments, Ant-v2 and Reacher-v2. The results are shown in Fig. 3. Fig. 3 (a) shows that the obsolescence technology helps relieve estimation bias, but the improvement is more pronounced after adding the Q-value weighting technology. Meanwhile, Fig. 3 (b) indicates that the Q-value weighting technology helps reduce the variance of estimation bias, allowing the estimation bias to converge faster. Fig. 3 shows that the estimation bias does not fluctuate even after eliminating the poor actors, indicating the Q-value weight still works and alleviates the estimation bias.

4. Experiment

4.1. Implementation details

To evaluate our method, we conduct extensive experiments on continuous control tasks from OpenAI Gym [30] simulated by MuJoCo. Table 2 indicates the dimensions of state and action. Fig. 4 shows the visual rendering of the environments. The darker lines represent the average return of 10 random seeds, and the shaded area represents the standard deviation.

We select the optimal parameters experimentally. Fig. 5 compares the performance of MA-TD3 with different weight coefficients η. The average return is the expected sum of rewards an agent obtains over time steps. It is calculated by taking the average of the returns obtained over 20 episodes. The return is sum of the agent's rewards over a single episode. The calculation of the average return remains consistent throughout the paper. Experiments show that the algorithm performs best in Ant-v2 and Walker2d-v2

Table 2
Details of 8 MuJoCo control tasks.

Environment	State Dimension	Action Dimension
Ant-v2	111	28
Hopper-v2	11	3
Halfcheetah-v2	17	6
Humannoid-v2	376	17
InvertedDoublePendulum-v2	11	1
InvertedPendulum-v2	4	1
Reacher-v2	11	2
Walker2d-v2	17	6

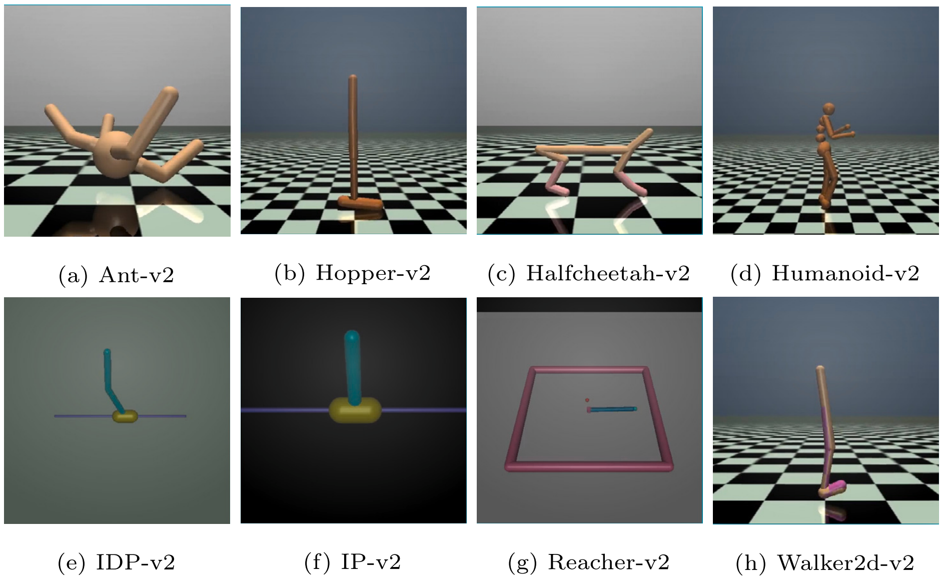


Fig. 4. Rendering of the MuJoCo continuous control tasks.

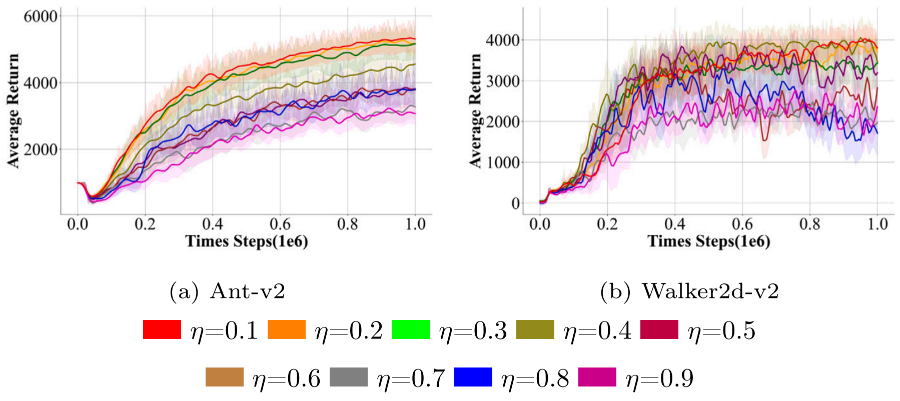
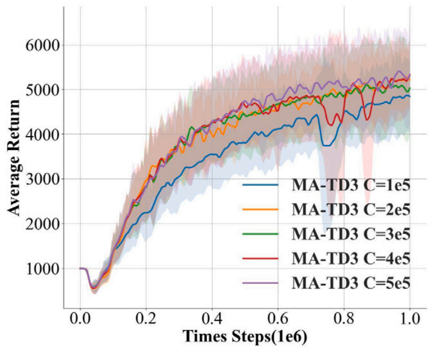


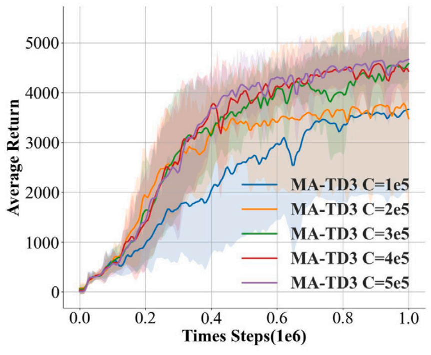
Fig. 5. Comparison of MA-TD3 different η in 2 MuJoCo environments.

environments when $\eta = 0.1$. As η increases, the algorithm suffers from severe underestimation problem leading to performance degradation. So, we apply $\eta = 0.1$ in later experiments.

Fig. 6 gives a comparative analysis of the algorithm's performance with different elimination steps. The two experiments show that the algorithm's performance improves with increasing elimination steps. However, increasing the elimination step means more significant computational effort, resulting in a longer computation time. Thus, we choose $C = 300,000$ in all environments to balance the algorithm performance and computation time. The hyper-parameters and settings of neural networks of the MA-TD3 are the same as that of the TD3. To balance algorithm performance and computational complexity, we set the number of actors K to 2, the number of environment M to 3, and the elimination step C to 300,000 in the MA-TD3. The weight coefficient η is 0.1 for all the



(a) Ant-v2



(b) Walker2d-v2

Fig. 6. Comparison of different elimination steps C in 2 MuJoCo environments.

Table 3
Hyperparameters comparison of MA-TD3 and TD3.

Hyperparameter	TD3	MA-TD3
Actor network	(256,256)	(256,256)
Critic network	(256,256)	(256,256)
Explore noise	0.1	0.1
Batch size	256	256
Discount	0.99	0.99
Optimizer	Adam	Adam
Actor learning rate	0.0003	0.0003
Critic learning rate	0.0003	0.0003
Target network update rate	0.005	0.005
Actor delay update frequency	2	2
Start time step	25,000	25,000
Actor number K	None	2
Number of environment M	None	3
Elimination step C	None	300,000
Competition frequency A	None	5,000
Weight coefficient η	None	0.1

Table 4
Comparison of max average return over 10 trials of 1 million time steps. ± corresponds to a single standard deviation over trials. The maximum value of 6 methods for each task is bolded.

Environment	MA-TD3	TADD	DARC	TD3	MA-DDPG	DDPG
Ant	**5512.1±807.1**	4579.0	5030.3	3443.5	1943.7	1258.7
Hopper	**3679.7±68.8**	3554.5	3454.6	3519.3	3562.8	3387.0
Halfcheetah	**12334.6±529.2**	10827.9	11572.1	9569.5	12075.9	10827.1
Humanoid	**5727.2±156.1**	5194.2	5308.6	5359.2	1897.5	1708.8
IDP	**9359.9±0.1**	8431.4	9352.6	8431.4	9354.4	8426.4
InvertedPendulum	**1000±0.0**	1000	1000	1000	1000	1000
Reacher	**-3.3±0.39**	-4.1	-4.1	-3.9	-3.5	-4.2
Walker2d	**4719.8±344.5**	4044.4	3251.8	4099.6	4045.4	2744.2

MuJoCo continuous control tasks. Each algorithm is run with 10 independent seeds and evaluated ten times every 5,000 timesteps. Table 3 illustrates other implementation details.

4.2. Experiment results

We compare the proposed algorithm with TD3, DDPG, TADD, and DARC. Fig. 7 shows the experimental results, where the darker lines represent the average return of 10 trials. The shaded area represents the 1 standard deviation. Table 4 shows the experimental results of the maximum average return over 10 trials of 1 million time steps.

The experimental results demonstrate that our algorithms significantly outperform the comparison algorithm in large state-action spaces environments Ant-v2, Hopper-v2, Humanoid-v2, Humanoid-v2, Walker2d-v2, and Halfcheetah-v2. The reason is that those environments with large state and action spaces are more complex and thus require powerful exploration capability and the ability to value accurately. In addition, our algorithm converges to the optimal strategy faster in all tasks, suggesting that the MAM mechanism

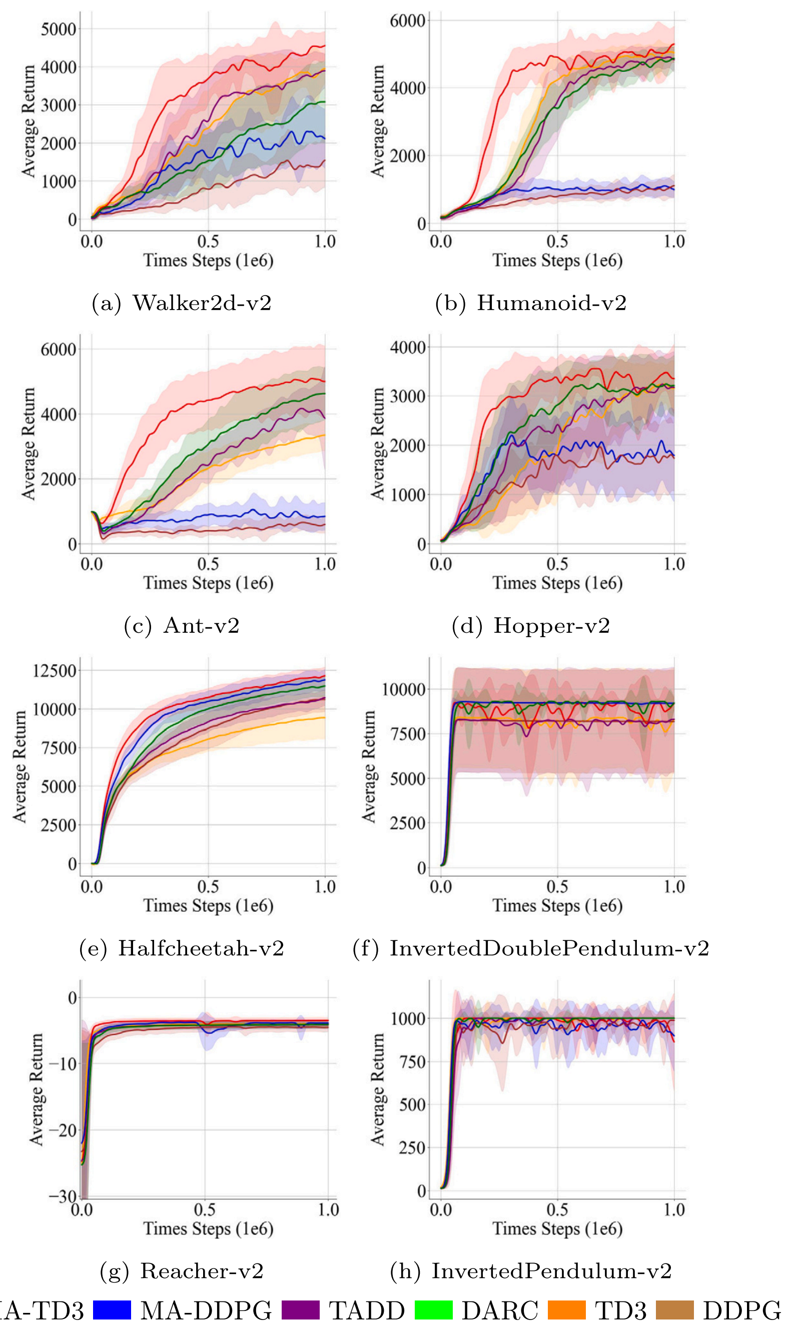


Fig. 7. Learning curves on 8 MuJoCo continuous control tasks.

can enhance exploration and generate high-quality experience, facilitating more stable and accurate value estimation in the early training stage.

To better verify the effectiveness of the obsolescence technology and Q-value weighting technology, we design ablation experiments in four classic MuJoCo continuous control environments, HalfCheetah-v2, Ant-v2, Hopper-v2, and Walker2d-v2. Fig. 8 shows the experimental results, which indicate that the algorithm's performance decreases significantly after the absence of the weighting technique, because the algorithm suffers from an underestimation problem, which makes the policy converge to a suboptimal policy. From the experiments, we find that poorly performing actors can hardly generate experience in the later stage of training. Thus, taking the obsolescence technology can prevent poorly performing actors from developing invalid experience, reducing computational costs and speeding up the training process. See Table 5.

Fig. 9 compares the convergence of MA-TD3 and TD3. In the early training stage, the standard deviation of reward returns is significant, indicating that the algorithm has not yet converged. The standard deviation gradually decreases as training proceeds, and

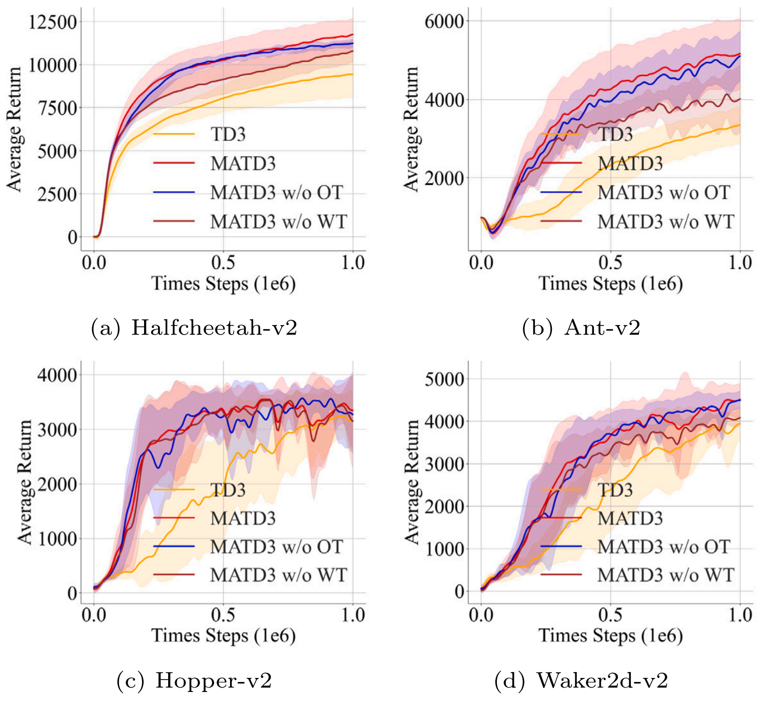


Fig. 8. Ablation experiments on Halfcheetah-v2 and Ant-v2, where w/o OT denotes without obsolescence technology, w/o WT represents without weighing technology.

Table 5
Ablation experiments results of max average return over 10 trials of 1 million time steps. The maximum value of 6 methods for each task is bolded.

Methods	HalfCheetah	Hopper	Walker2d	Ant
MA-TD3	**12334.6**	**3679.7**	**4719.8**	**5515.1**
DARC	11572.1	3454.6	3251.8	5030.3
TADD	10827.9	3554.5	4044.4	4579.0
TD3	9569.5	3519.3	4099.6	3443.5
MA-TD3 - OT	11558.2	3553.9	4615.0	5324.2
MA-TD3 - WT	10755.3	3531.0	4240.9	4327.8

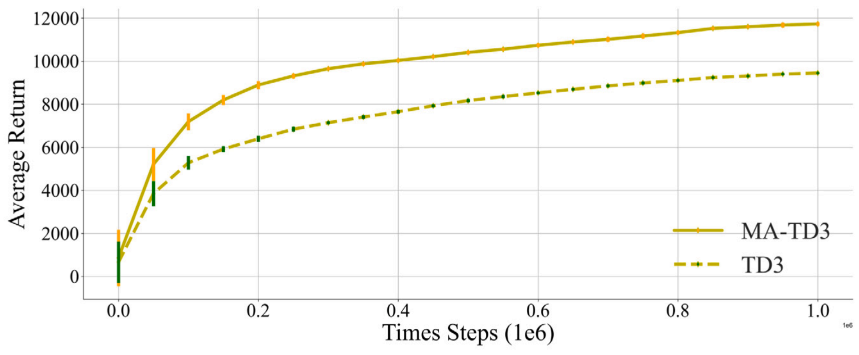
both algorithms converge to their best performance. The experimental results show that embedding MAM ensures a higher return convergence.

5. Conclusion

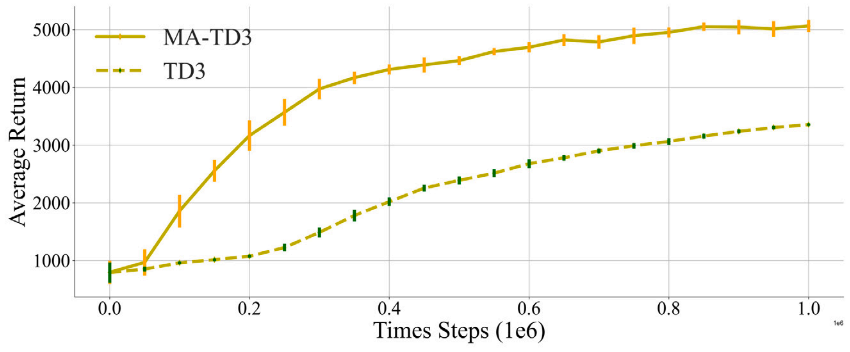
This paper proposes a multi-actor mechanism (MAM) incorporating both obsolescence technology and Q-value weighting technology to exploit the multi-actor advantages. The obsolescence technology can enhance the exploration capability of the agent, and the Q-value weighting technology can reduce the estimation bias and variance to achieve a more accurate value estimation. To illustrate the advantages of MAM, we embed MAM into DDPG to construct the algorithm MA-DDPG and verify in a artificial experiment that the obsolescence technology is beneficial for enhancing exploration ability. Then, we offer the MA-TD3 by embedding MAM into the TD3 and demonstrate that MAM can reduce the estimation bias and variance through both experiments and theory. Extensive experiments illustrate that MA-TD3 outperforms state-of-the-art methods. Interesting directions for future work include exploring how to take advantage of multi-actor with different characteristics to adapt to more complex environments and improve the RL algorithm's performance.

CRediT authorship contribution statement

Lin Li: Conceptualization, Methodology, Software, Writing – review & editing. **Yuze Li:** Methodology, Software, Writing – original draft. **Wei Wei:** Project administration, Supervision, Writing – review & editing. **Yujia Zhang:** Software, Writing – review & editing. **Jiye Liang:** Methodology, Project administration, Writing – review & editing.



(a) HalfCheetah-v2



(b) Ant-v2

Fig. 9. Convergence experiments on Halfcheetah-v2 and Ant-v2, where the error bars correspond to standard deviation during evaluation with 10 random seeds.

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

Data will be made available on request.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No.2019YFE0118200), the National Natural Science Foundation of China (Nos.62276160, 61976184, 61772323), the Natural Science Foundation of Shanxi Province, China (No.202203021211291), and the 1331 Engineering Project of Shanxi Province (First-class Discipline Construction Project of Computer Science and Technology), China.

References

[1] R.S. Sutton, Learning to predict by the methods of temporal differences, Mach. Learn. 3 (1) (1988) 9–44.
[2] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, J. Morimoto, Deep learning, reinforcement learning, and world models, Neural Netw. 152 (2022) 267–275.
[3] X. Wang, T. Sandholm, Reinforcement learning to play an optimal Nash equilibrium in team Markov games, in: NIPS, 2002, pp. 1603–1610.
[4] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., Grandmaster level in starcraft ii using multi-agent reinforcement learning, Nature 575 (7782) (2019) 350–354.
[5] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[6] Z. Bing, C. Lemke, L. Cheng, K. Huang, A. Knoll, Energy-efficient and damage-recovery slithering gait design for a snake-like robot based on reinforcement learning and inverse reinforcement learning, Neural Netw. 129 (2020) 323–333.
[7] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, G. Cheng, Learning cpg-based biped locomotion with a policy gradient method: application to a humanoid robot, Int. J. Robot. Res. 27 (2) (2008) 213–228.

[8] A. Tsantekidis, N. Passalis, A. Tefas, Diversity-driven knowledge distillation for financial trading using deep reinforcement learning, Neural Netw. 140 (2021) 193–202.
[9] Y. Lin, F. Lin, L. Yang, W. Zeng, Y. Liu, P. Wu, Context-aware reinforcement learning for course recommendation, Appl. Soft Comput. (2022) 109189.
[10] F. Liu, R. Tang, H. Guo, X. Li, Y. Ye, X. He, Top-aware reinforcement learning based recommendation, Neurocomputing 417 (2020) 255–269.
[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.
[12] S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in: Proceedings of the 1993 Connectionist Models Summer School, vol. 6, 1993.
[13] H. Hasselt, Double q-learning, in: NIPS, vol. 23, 2010.
[14] R. Fox, A. Pakman, N. Tishby, Taming the noise in reinforcement learning via soft updates, arXiv preprint, arXiv:1512.08562, 2015.
[15] D. Lee, B. Defourny, W.B. Powell, Bias-corrected q-learning to control max-operator bias in q-learning, in: ADPRL, 2013, pp. 93–99.
[16] O. Nachum, M. Norouzi, G. Tucker, D. Schuurmans, Smoothed action value functions for learning Gaussian policies, in: ICML, 2018, pp. 3692–3700.
[17] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep reinforcement learning, in: ICML, 2016, pp. 1995–2003.
[18] H. v. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: AAAI, 2016, pp. 2094–2100.
[19] O. Anschel, N. Baram, N. Shimkin, Averaged-dqn: variance reduction and stabilization for deep reinforcement learning, in: ICML, 2017, pp. 176–185.
[20] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint, arXiv:1509.02971, 2015.
[21] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: ICML, 2018, pp. 1587–1596.
[22] D. Wu, X. Dong, J. Shen, S.C. Hoi, Reducing estimation bias via triplet-average deep deterministic policy gradient, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2020) 4933–4945.
[23] Q. He, X. Hou, Reducing estimation bias via weighted delayed deep deterministic policy gradient, arXiv preprint, arXiv:2006.12622, 2020.
[24] L. Pan, Q. Cai, L. Huang, Softmax deep double deterministic policy gradients, in: NIPS, vol. 33, 2020, pp. 11767–11777.
[25] W. Wei, Y. Zhang, J. Liang, L. Li, Y. Li, Controlling underestimation bias in reinforcement learning via quasi-median operation, in: AAAI, vol. 36, 2022, pp. 8621–8628.
[26] J. Lyu, X. Ma, J. Yan, X. Li, Efficient continuous control with double actors and regularized critics, in: AAAI, vol. 36, 2022, pp. 7655–7663.
[27] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, K. Tumer, Collaborative evolutionary reinforcement learning, in: ICML, PMLR, 2019, pp. 3341–3350.
[28] E. Todorov, T. Erez, Y. Tassa, Mujoco: a physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.
[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: ICML, 2014, pp. 387–395.
[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint, arXiv:1606.01540, 2016.