Chinese Journal of Aeronautics, (2024), **xxx**(xx): xxx–xxx

Chinese Society of Aeronautics and Astronautics
& Beihang University
Chinese Journal of Aeronautics
cja@buaa.edu.cn
www.sciencedirect.com

FULL LENGTH ARTICLE

Controlling underestimation bias in reinforcement learning via minmax operation

Fanghui HUANG [a,b], Yixin HE [b], Yu ZHANG [a], Xinyang DENG [a], Wen JIANG [a,*]

[a] *School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China*
[b] *College of Information Science and Engineering, Jiaxing University, Jiaxing 314001, China*

Received 20 July 2023; revised 6 March 2024; accepted 6 March 2024

KEYWORDS

Reinforcement learning;
Minmax operation;
Estimation bias;
Underestimation bias;
Variance

Abstract Obtaining the accurate value estimation and reducing the estimation bias are the key issues in reinforcement learning. However, current methods that address the overestimation problem tend to introduce underestimation, which face a challenge of precise decision-making in many fields. To address this issue, we conduct a theoretical analysis of the underestimation bias and propose the minmax operation, which allow for flexible control of the estimation bias. Specifically, we select the maximum value of each action from multiple parallel state-action networks to create a new state-action value sequence. Then, a minimum value is selected to obtain more accurate value estimations. Moreover, based on the minmax operation, we propose two novel algorithms by combining Deep Q-Network (DQN) and Double DQN (DDQN), named minmax-DQN and minmax-DDQN. Meanwhile, we conduct theoretical analyses of the estimation bias and variance caused by our proposed minmax operation, which show that this operation significantly improves both underestimation and overestimation biases and leads to the unbiased estimation. Furthermore, the variance is also reduced, which is helpful to improve the network training stability. Finally, we conduct numerous comparative experiments in various environments, which empirically demonstrate the superiority of our method.

1. Introduction

Reinforcement Learning (RL) aims to find optimal policies for sequential decision-making problems by allowing the agent to interact with the environment.[1] In recent years, many algorithms, such as Q-learning and policy gradient, have been proposed to address some challenges.[2] Currently, RL can solve many previously intractable problems by mimicking human behavior. For example, RL has been used for precision

* Corresponding author.
E-mail address: jiangwen@nwpu.edu.cn (W. JIANG).
☆ Peer review under responsibility of Editorial Committee of CJA.

Production and hosting by Elsevier

https://doi.org/10.1016/j.cja.2024.03.008
1000-9361

decision-making in aerospace,[3] energy management,[4] wireless communications[5] and other fields.

In particular, the application of RL in the field of aeronautics and astronautics has been rapidly developed, such as space operations, aircraft trajectory planning and positioning. Specifically, in the air traffic management, for the demand and capacity balancing problem, Chen et al.[6] defined multiple flights as a multi-agent component partial observation multi-agent decision-making system. A cooperation coefficient was introduced into the reward function to control the intelligent decision of air traffic. The trajectory efficiency of the robot using a wrench to insert a hex hole into a bolt still needs to be improved. Zhang et al.[7] proposed the improved trajectory efficiency method based on RL, which was modelled as an objective optimization problem to enhance the trajectory efficiency. In terms of trajectory optimization of solar-powered Unmanned Aircraft Vehicles (UAV), Ni et al.[8] designed a solar-powered aircraft trajectory optimization and guidance based on RL framework, which could achieve an autonomous flight based on energy maximization. Aiming at the problem of high-precision positioning in UAV formations, Ma et al.[9] presented a UAV formation control algorithm based on RL to achieve the high efficiency and robustness of the controller, and obtained high-precision positioning data. We can see that RL has penetrated into various fields of aeronautics and astronautics. In these fields, high precision must be considered. Therefore, the need for accuracy requires more attention in the application of RL. However, the presence of estimation bias affects the performance of RL algorithms, making it difficult to obtain high- quality decisions.

The estimation bias problem is caused by inaccurate or biased estimation values, which is a critical issue in RL.[10] Research has shown that the estimation bias will lead to suboptimal policies, degraded performance, and limited application range. Overestimation is a prevalent form of the estimation bias, particularly in classical RL algorithms, such as Q-learning and its variants. This bias is mainly caused by the use of max operator and imprecise estimation of the state-action function (Q function).[11] With the increasing complexity of the environments, the classical RL algorithms can no longer meet the growing demands. To address this issue, Deep RL (DRL) algorithms using multi-layer neural networks have emerged, such as Deep Q-Network (DQN).[12] However, the use of neural networks to approximate the Q function provides an additional source of the estimation bias, further exacerbating the overestimation problem. In addition, overestimation can also be amplified over time, leading to the positive bias that prevents the agent from making efficient decisions in many complex tasks.[13]

Recently, several techniques have been proposed to address the problem of estimation bias in RL, such as regularization, weighted average estimation, decoupling operation, and min operator. For instance, Cheng et al.[14] proposed an improved Q-learning method that controlled the regularization policy based on prior information to reduce the bias in the generalized RL. Cicek et al.[15] solved the estimation bias problem in the continuous control tasks by using the experience replay mechanism. To improve the accuracy of Q estimation, Fox et al.[16] designed the G-learning algorithm, which regularized the estimated values by punishing deterministic policies during the initial stage of agent learning. To alleviate the value estimation uncertainty among double critic networks, Lv et al.[17] introduced Double Actors Regularized Critics (DARC) by adding critical regularization. Moreover, Hassam et al.[18] added diversity to the loss function. Next, Gao et al.[19] proposed the Q-ensemble algorithm based on diversity, which improved the sampling efficiency and reduced overestimation. However, these methods require prior information, which can be difficult to obtain and may be inaccurate and subjective. In addition, the regularization methods lack theoretical support and mainly rely on empirical analysis.

Furthermore, Anschel et al.[20] proposed two algorithms, Averaged-DQN and Ensemble-DQN, to reduce the estimation bias and variance. The Averaged-DQN algorithm averaged multiple state-action values (Q values) to provide more accurate estimations. The Ensemble-DQN algorithm applied the idea of ensemble learning. Subsequently, Lee et al.[21] presented the Sunrise framework, which combined weighted Bellman backups with multiple networks to solve the target network error problem. Cini et al.[22] proposed the weighted Q-learning algorithm that calculated the probability of different actions based on Gaussian sampling and used the weighted sum of diverse actions to mitigate overestimation. Similarly, Song et al.[23] adopted the softmax operation to weight multiple estimated Q values for better accuracy. However, for a finite number of Q-networks, these operations can only alleviate overestimation and not completely eliminate it. In addition, the settings of various actions probabilities are relatively ideal and have poor environmental applicability.

To address the above challenges, various methods based on decoupling operation or min operator have been proposed. Decoupling operation decouples the calculation of estimated Q values and the selection actions to eliminate overestimation. One example was the double Q-learning algorithm designed by Van Hasselt,[24] which used two independent Q estimators to calculate Q values and select actions, respectively. Building on this work, Van Hasselt et al.[25] proposed Double DQN (DDQN) by combining double Q-learning with multi-layer neural networks. However, the above decoupling operations solve the overestimation problem on discrete control tasks and introduce the underestimation bias. To address this issue, Twin Delayed Deep Deterministic policy gradient (TD3)[26] was presented, which minimized the values of two critic networks and adopted the delayed policy update to reduce error propagation. Similarly, Lan et al.[27] proposed the Maxmin Q-learning, which used the min operator to obtain the minimum value of multiple parallel Q-networks as the estimated Q value. Li et al.[28] calculated the minimum and maximum values of two critic networks by using the min and max operators, respectively, and controlled the estimation bias by adding the min and max values. Subsequently, Zhang et al.[29] designed a weighted double Q-learning method to adjust overestimation by assigning different weights to the underestimation caused by the min operator and the overestimation caused by the max operator. Karimpanal et al.[30] set adaptive parameters to adjust the weights, which could control the bias and improve the applicability of the algorithms. However, the above methods often result in underestimation bias and have negative impact on the systems.[31] In addition, the weights are too simple and fixed, and often set for specific environments. If the environment is changed, the performance of these algorithms may be significantly affected.

The above methods have mitigated the overestimation bias, but also have brought the underestimation bias. The min oper-

Please cite this article in press as: HUANG F et al. Controlling underestimation bias in reinforcement learning via minmax operation, *Chin J Aeronaut* (2024), https://doi.org/10.1016/j.cja.2024.03.008

ator is widely used to control the estimation bias. However, most methods analyze the performance of min operator empirically, and lack theoretical analysis. Moreover, these methods mainly focus on overestimation and ignore underestimation. Therefore, there are three challenging issues that need to be considered and addressed: (A) how to measure the degree of underestimation; (B) how to design a method to control the estimation bias; (C) how to conduct theoretical analysis to evaluate the performance of the proposed method in estimating bias.

Based on the challenges and issues discussed above, we theoretically derive the closed-form expression of underestimation caused by the min operator. In addition, we propose the minmax operation by selecting the maximum value from multiple parallel Q-networks, which mitigates the underestimation bias and controls the estimation bias. Moreover, we design two novel algorithms, named minmax-DQN and minmax-DDQN, to improve the applicability of minmax operation to the environments. Furthermore, the performance of minmax operation is theoretically analyzed. Theoretical analysis shows that the minmax operation alleviates the underestimation bias, and effectively controls the estimation bias, which can achieve the switch among underestimation, overestimation and unbiased estimation. In addition, we also derive the closed-form expression of variance by using the minmax operation, which can effectively reduce the variance and improve the network training stability. Finally, simulation results demonstrate the superiority of our proposed algorithms in various environments.

The main contributions of this paper are summarized as follows:

(1) We analyze the sources of underestimation bias and theoretically derive the underestimation caused by the min operator. As a solution, we propose the minmax operation by selecting the maximum value from multiple Q-networks, which can mitigate the underestimation, and effectively control the estimation bias and variance.
(2) Combining the minmax operation with typical RL algorithms (i.e., DQN and DDQN), we propose two novel algorithms, named minmax-DQN and minmax-DDQN. In this way, the adaptability and robustness of the minmax operation to different environments can be improved.
(3) We theoretically derive the closed-form expressions of minmax operation about controlling estimation bias and variance, demonstrating that it can achieve unbiased estimation, as well as the transformation between underestimation and overestimation. Moreover, the variance can be significantly reduced, thereby improving network training performance.

The rest of this paper is organized as follows. In Section 2, we provide a brief introduction to the relevant theories of RL. Section 3 gives theoretical analysis of underestimation caused by min operator. In Section 4, we introduce our proposed minmax operation, and describe the two novel algorithms, as well as the closed-form expressions of the bias and variance. Several environmental results are given in Section 5 to show the effectiveness of our method. We draw a conclusion in Section 6.

2. Preliminaries

RL is a branch of machine learning that simulates the human thinking, which is mainly used to describe and solve the problem of learning policies to maximize the returns in the process of interacting with the environment.[32,33] The general RL framework can be modelled in the form of a Markov Decision Process (MDP),[34] expressed as (S, A, r, γ, P). In this paper, we consider the finite MDP, where S and A represent the spaces of state and action, respectively; γ is the discount factor; r is the reward function; P is the state transition probability. At time l, in a state $s_l \in S$, the agent selects and performs an action $a_l \in A$. At this point, the agent receives an immediate reward and transitions to a next state $s_{l+1} \in S$. The cumulative discount reward from time l is denoted as the return $G_l = \sum_{l\prime=l}^{L} \gamma^{l\prime - l} r_{l\prime}$.[35] The agent's objective is to find the optimal policy πthat maximizes the expected return $\mathbb{E}_\pi[G_l]$, where $l\prime$ is a variable, and L is the total time step.

Value-based RL algorithms encode the policy by using the Q function, that is, $Q^\pi(s,a) = \mathbb{E}_\pi[G_l]$. The Q values are updated through the Bellman equation, which provides a calculated way for the state transition process $(s_l, a_l, r_l, s\prime)$ of the agent, as shown below.[36]

$$Q^\pi(s,a) = r_l + \gamma \mathbb{E}_{a\in A}[Q^\pi(s\prime, a)] \tag{1}$$

Currently, DRL algorithms are mainly used to deal with environments with a large number of state-action pairs. Popular DRL algorithms, such as DQN and DDQN, approximate the Q function by using neural networks. In addition, the Q network parameter $\theta$is updated by minimizing the loss function.[37]

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a\prime \in A} Q(s\prime, a\prime; \theta\prime) - Q(s,a;\theta)\right)^2\right] \tag{2}$$

where $\theta\prime$ is the parameter of the target network.

3. Underestimation bias using min operator

Typical RL algorithms use the max operator to estimate Q values. However, research has shown that the max operator can cause overestimation bias.[38] To address this issue, using the min operator instead of the max operator has been proposed. Unfortunately, using the min operator will lead to underestimation bias, which has primarily been analyzed empirically.[39] At present, the theoretical analysis of this problem is still lacking. Therefore, we carry out a theoretical derivation of the estimation bias when using the min operator.

When the max operator is adopted in RL, the Q target is typically calculated as $Q^T(s,a) = r + \gamma \max_a Q(s,a)$. Similarly, when the min operator is adopted, the Q target is calculated as $Q^T(s,a) = r + \gamma \min_a Q(s,a)$. According to Thrun and Schwartz,[38] using a single estimator $Q(s,a)$ with the min operator will lead to the estimation bias $Z_d(s,a)$, which can be defined as

$$Z_d(s,a) = \gamma\left(\min_a Q(s,a) - \min_a Q^{\mathrm{true}}(s,a)\right) \tag{3}$$

where $Q(s,a) = Q^{\text{true}}(s,a) + E_d(s,a)$, $Q(s,a)$ and $Q^{\text{true}}(s,a)$ are the estimated and true Q values, respectively, and $E_d(s,a)$ represents random approximation error, which is a random variable following a uniform distribution $U(-\varsigma,\varsigma)$ for $\varsigma > 0$.

In the following, we derive the expectation of the estimation bias in Theorem 1.

Theorem 1. We assume that there are m actions available in state s, and then the expected estimation bias by using the min operator with a single estimator $Q(s,a)$ can be expressed as

$$\mathbb{E}[Z_d(s,a)] = \gamma\varsigma\frac{1-m}{m+1} \tag{4}$$

where γ is the discount factor, and $\gamma \in [0,1]$.

Proof (Proof of Theorem 1).Owing to $E(s,a) \sim U(-\varsigma,\varsigma)$, the Probability Density Function (PDF) $f(x)$ of $E_d(s,a)$ is represented as $f(x) = \begin{cases} \frac{1}{2\varsigma}, -\varsigma \leqslant x \leqslant \varsigma \\ 0, \text{else} \end{cases}$ and the Cumulative Distribution Function (CDF) $F(x)$ of $E_d(s,a)$ is represented as $F(x) = \begin{cases} 0, x < -\varsigma \\ \frac{1}{2} + \frac{x}{2\varsigma}, -\varsigma \leqslant e \leqslant \varsigma \\ 1, x > \varsigma \end{cases}$, where ς is a constant and $\varsigma > 0$.

Based on the definition of mathematic expectation, we have

$$\begin{aligned} \mathbb{E}[Z_d(s,a)] = & \ \mathbb{E}\left[\gamma\left(\min_a Q(s,a) - \min_a Q^{\text{true}}(s,a)\right)\right] \\ = & \ \mathbb{E}\left[\gamma \min_a E_d(s,a)\right] \\ = & \ \gamma \int_{-\varsigma}^{\varsigma} mxf(x)[1-F(x)]^{m-1}\mathrm{d}x \\ = & \ \gamma \int_{-\varsigma}^{\varsigma} \frac{mx}{2\varsigma}\left(\frac{1}{2} - \frac{x}{2\varsigma}\right)^{m-1}\mathrm{d}x \\ = & \ -\gamma \int_{-\varsigma}^{\varsigma} x\mathrm{d}\left(\frac{1}{2} - \frac{x}{2\varsigma}\right)^{m} \\ = & \ -\gamma x\left(\frac{1}{2} - \frac{x}{2\varsigma}\right)^{m}\Big|_{-\varsigma}^{\varsigma} + \gamma \int_{-\varsigma}^{\varsigma}\left(\frac{1}{2} - \frac{x}{2\varsigma}\right)^{m}\mathrm{d}x \end{aligned} \tag{5}$$

$$\begin{aligned} = & \ -\gamma\varsigma + \gamma\int_{-\varsigma}^{\varsigma}\left(\frac{1}{2} - \frac{x}{2\varsigma}\right)^{m}\mathrm{d}x \\ = & \ -\gamma\varsigma + 2\gamma\varsigma\int_0^1 z^m \mathrm{d}z \quad \left(z \overset{\text{def}}{=} \frac{1}{2} - \frac{x}{2\varsigma}\right) \\ = & \ -\gamma\varsigma + 2\gamma\varsigma\frac{z^{m+1}}{m+1}\Big|_{-\varsigma}^{\varsigma} \\ = & \ \gamma\varsigma\frac{1-m}{m+1} \end{aligned}$$

Therefore, Theorem 1 is proved. ■

From Eq. (5), we can get the following conclusions:(1) When $m = 1$, we can easily obtain $\mathbb{E}[Z_d(s,a)] = 0$. In this situation, the estimation bias is 0.(2) When $m > 1$, we can easily obtain $\mathbb{E}[Z_d(s,a)] < 0$. In this situation, the expected estimation bias is negative, i.e., the underestimation bias exists.

In general, for all values of m, the values of $\mathbb{E}[Z_d(s,a)]$ are always less than or equal to 0, i.e., $\mathbb{E}[Z_d(s,a)] \leqslant 0$. However, in practical environments, the number of actions m that the agent can perform is usually greater than 1. Therefore, it can be observed that the underestimation bias always exists when using the min operator.

4. Proposed method

In this paper, we propose the minmax operation, which effectively addresses the problem of underestimation caused by the min operator. To illustrate the adaptability of minmax operation, we integrate it into DQN and DDQN algorithms, and present two novel algorithms, named minmax-DQN and minmax-DDQN. Moreover, we derive the closed-form expressions theoretically, which can prove the superiority of the minmax operation in controlling the estimation bias and reducing the variance. Furthermore, the minmax operation can improve the network training efficiency.

4.1. Minmax operation

To obtain accurate Q values, average and max operations are usually used, but they often lead to overestimation bias. In order to solve the overestimation problem, the min operator is introduced. Unfortunately, the underestimation phenomenon is caused. In light of these challenges, we propose a novel method named the minmax operation, which effectively controls the estimation bias, as well as reduces the variance. The framework of our proposed minmax operation is illustrated in Fig. 1.

Specifically, let $Q_i^{\text{est}}(s,a)$ denote the i-th Q values from Q networks. There are n independent $Q_i^{\text{est}}(s,a)$ from different Q networks for $i = 1,2,\cdots,n$. Our idea is to maintain n estimates of Q networks, and select the maximum of these estimates, denoted as $\max_{i\in\{1,2,\cdots,n\}} Q_i^{\text{est}}(s,a)$, which contains the Q values for all selectable actions a. Next, we select the minimum Q values from $\max_{i\in\{1,2,\cdots,n\}} Q_i^{\text{est}}(s,a)$ as the estimated Q value, denoted as $\min_a \max_{i\in\{1,2,\cdots,n\}} Q_i^{\text{est}}(s,a)$. In other words, $\min_a \max_{i\in\{1,2,\cdots,n\}} Q_i^{\text{est}}(s,a)$ is the minimum value among the maximum Q values obtained from the n estimates. This approach allows us to control the estimation bias and reduce the variance. Formally, the minmax operation can be defined as

$$Q^{\text{est}}(s,a) = \min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\text{est}}(s,a) \tag{6}$$

where $Q_i^{\text{est}}(s,a)$ is the estimated value of Q network.

Obviously, when $n = 1$, the minmax operation is equal to the min operator, resulting in an underestimation. It is observed that our proposed minmax operation is extensible.

The minmax operation can be integrated into any model-free RL algorithms, because it only modifies the estimation of Q values without altering the framework. To better demonstrate the superior performance of the minmax operation, we design two novel algorithms, named minmax-DQN and minmax-DDQN. Specifically, the minmax-DQN algorithm combines the minmax operation with the classic DQN to approximate Q values through the network. The target network is updated by the following loss function:

$$L_i^{\text{minmax-DQN}}(\theta) = \mathbb{E}\left[\left(y^{\text{DQN}} - Q_i^{\text{est}}(s\prime,a;\theta)\right)^2\right] \tag{7}$$

where $\theta\prime$ is the parameter of target network, θ is the parameter of Q network, and y^{DQN} represents the target value, $y^{\text{DQN}} = r + \gamma\min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\text{est}}(s\prime,a;\theta\prime)$.

The pseudo code of the minmax-DQN is shown as Algorithm 1.

To further illustrate the validity and applicability of our proposed minmax operation, we also combine the minmax operation with DDQN to design the minmax-DDQN algorithm. y^{DQN} is replaced with y^{DDQN}, and

Please cite this article in press as: HUANG F et al. Controlling underestimation bias in reinforcement learning via minmax operation, *Chin J Aeronaut* (2024), https://doi.org/10.1016/j.cja.2024.03.008

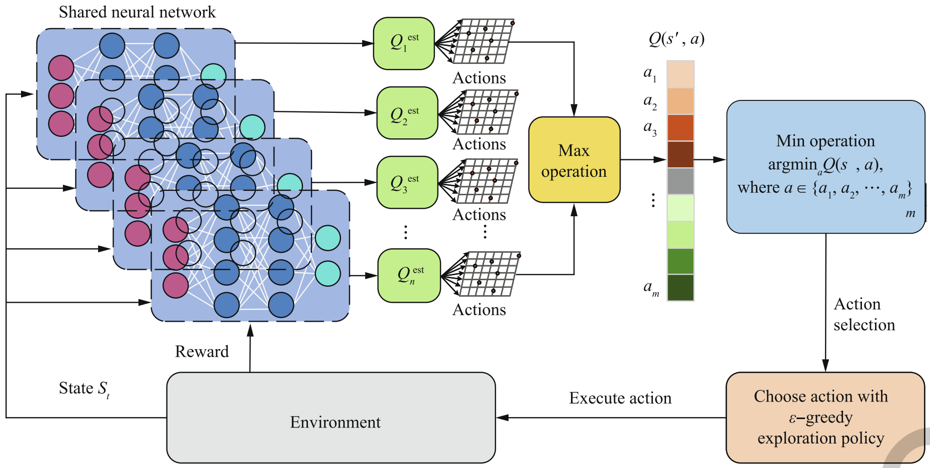

Fig. 1 Framework of minmax operation.

$y^{\mathrm{DDQN}} = r + \gamma Q_i^{\mathrm{est}}(s\prime, \arg\min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s\prime, a; \theta); \theta\prime)$. Therefore, the loss function can be expressed as

$$L_i^{\mathrm{minmax-DDQN}}(\theta) = \mathbb{E}\left[\left(y^{\mathrm{DDQN}} - Q_i^{\mathrm{est}}(s\prime, a; \theta)\right)^2\right] \tag{8}$$

4.2. Control the estimation bias via minmax operation

In this section, we demonstrate the superiority of the minmax operation in the estimation bias of Q values. First, we describe the meaning of estimation bias. Subsequently, we analyze the distribution of the maximum and minimum values of the sequence through Lemma 1. Moreover, based on Lemma 1, we provide a theoretical analysis of the expectation of estimation bias by using the minmax operation, and present the result in Theorem 2. Furthermore, we provide an in-depth analysis and explanation of the estimation bias expectation results. Our analysis clearly demonstrates the ability of the minmax operation to effectively control the estimation bias, both in terms of overestimation and underestimation.

Algorithm 1. Minmax-DQN.

1: Initialize Q networks $Q_i^{\mathrm{est}}(s\prime, a; \theta)$, target Q networks $Q_i^{\mathrm{est}}(s\prime, a; \theta\prime)$ for $i = 1, 2, \cdots, n$, and replay memory D.
2: **while** agent interact with the environment **do**
3: Calculate $Q^{\mathrm{est}}(s, a) = \min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s, a)$.
4: Select action a by ε-greedy policy based on $Q^{\mathrm{est}}(s, a)$.
5: Observe r and next state $s\prime$.
6: Store transition $(s, a, r, s\prime)$ in $D(s, a, r, s\prime)$.
7: **for** $i = 1, 2, \cdots, n$ **do**
8: Sample a mini-batch $(s, a, r, s\prime) \sim D$.
9: Update target: $y = r + \gamma \min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s\prime, a; \theta\prime)$.
10: Using $L_i(\theta) = \mathbb{E}[(r + \gamma \min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s\prime, a; \theta\prime) - Q_i^{\mathrm{est}}(s\prime, a; \theta))^2]$ to update $\theta\prime$.
11: **end for**
12: **end while**

The concept of estimation bias was proposed by Thrun and Schwartz.[38] Following their conventions, we denote $Q_i^{\mathrm{est}}(s, a)$ and $Q^{\mathrm{true}}(s, a)$ as the estimated Q value and true Q value, respectively. Then, the estimation bias can be expressed as follows:

$$Z(s, a) = \gamma\left(\min_a \max_{i\in\{1,2,\cdots,n\}} Q_i^{\mathrm{est}}(s, a) - \min_a Q^{\mathrm{true}}(s, a)\right) \tag{9}$$

where $\max_{i\in\{1,2,\cdots,n\}} Q_i^{\mathrm{est}}(s, a) = Q^{\mathrm{true}}(s, a) + \max_{i\in\{1,2,\cdots,n\}} E_i(s, a)$, where $E_i(s, a)$ represents random approximation error, which is a random variable following uniform distribution $U(-\varsigma, \varsigma)$ for $\varsigma > 0$, and n is the number of Q networks.

It can be seen that each $Q_i^{\mathrm{est}}(s, a)$ has a $E_i(s, a)$, and $E_i(s, a)$ can be positive or negative. When $E_i(s, a) = 0$, it indicates that the estimation of Q value is accurate. In this case, the value of estimation bias is 0, where $Q_i^{\mathrm{est}}(s, a) = Q^{\mathrm{true}}(s, a)$.

Experience shows that whether the estimation bias $Z(s, a)$ is positive or negative (i.e., the overestimation or underestimation), it will cause a certain degree of negative impact on most systems.[40] Although overestimation or underestimation may have a positive effect in some cases, it cannot guarantee the optimal system performance. Therefore, we need a method that can minimize the underestimation bias and even control bias.

In the following, we analyze the distribution of random approximation errors $E(s, a)$ by using Lemma 1. This analysis will aid in deriving the expected estimation bias and variance, as presented in Theorem 2 and Theorem 3, respectively.

Lemma 1. We assume $E(s, a) = \{E_1(s, a), E_2(s, a), \cdots, E_n(s, a)\} \sim U(-\varsigma, \varsigma)$ with PDF $f(x)$ and CDF $F(x)$. We define $E_{\min} = \min_{i\in\{1,2,\cdots,n\}} E_i(s, a)$ with PDF $f_{\min}(x)$ and CDF $F_{\min}(x)$, respectively, and $E_{\max} = \max_{i\in\{1,2,\cdots,n\}} E_i(s, a)$ with PDF $f_{\max}(x)$ and CDF $F_{\max}(x)$, respectively. Then, we have(1) $F_{\min}(x) = 1 - [1 - F(x)]^n$, and $f_{\min}(x) = nf(x)[1 - F(x)]^{n-1}$.(2) $F_{\max}(x) = [F(x)]^n$, and $f_{\max}(x) = nf(x)[F(x)]^{n-1}$.

Please cite this article in press as: HUANG F et al. Controlling underestimation bias in reinforcement learning via minmax operation, *Chin J Aeronaut* (2024), https://doi.org/10.1016/j.cja.2024.03.008

Proof (Proof of Lemma 1).

(1) According to the definition of CDF and $E_{\min}$, we know $F_{\min}(x) = P(E_{\min} x) = 1 - P(E_{\min} > x) = 1 - P(E_1 > x, E_2 > x, \cdots, E_n > x) = 1 - P(E_1 > x), \cdots, \ P(E_n > x) = 1 - [1 - F(x)]^n$. Then, we can obtain the PDF of $E_{\min}$, which is expressed as $f_{\min}(x) = \frac{dF_{\min}(x)}{dx} = nf(x)[1 - F(x)]^{n-1}$.

(2) Similar to (1), we can obtain the CDF of $E_{\max}$, which is expressed as $F_{\max}(x) = P(E_{\max} \leqslant x) = P(E_1 \leqslant x, E_2 \leqslant x, \cdots, E_n \leqslant x) = P(E_1 \leqslant x) \cdots P(E_n \leqslant x) = [F(x)]^n$. Then, the PDF of $E_{\max}$ is calculated as $f_{\max}(x) = \frac{dF_{\max}(x)}{dx} = nf(x)[F(x)]^{n-1}$. ■.

Subsequently, according to Theorem 2, we theoretically derive the expected estimation bias $\mathbb{E}[Z(s,a)]$ by using the minmax operation. In addition, Theorem 2 also describes the relationship between the expected estimation bias and the number of Q networks n.

Theorem 2. We assume that the number of available actions in state s is m. Under the condition that $E(s,a) = \{E_1(s,a), E_2(s,a), \cdots, E_n(s,a)\}$obeys $U(-\varsigma, \varsigma)$, the expected estimation bias of $\min\limits_{a} \max\limits_{i \in \{1,2,\cdots,n\}} Q_i^{\text{est}}(s,a)$ is

$$\mathbb{E}[Z(s,a)] = \gamma\varsigma\left[\frac{2\Gamma(m+1)\Gamma\left(\frac{1}{n}\right)}{n\Gamma\left(m+\frac{1}{n}+1\right)} - 1\right] \tag{10}$$

Proof (Proof of Theorem 2).

Since $E(s,a) = \{E_1(s,a), E_2(s,a), \cdots, E_n(s,a)\}$ obeys $U(-\varsigma, \varsigma)$, we can easily obtain the PDF $f(x)$ and CDF $F(x)$ of $E_i(s,a)$, which are $f(x) = \frac{1}{2\varsigma}$ and $F(x) = \frac{1}{2} + \frac{x}{2\varsigma}$, respectively. In addition, combining with the results of Lemma 1, we know that the expected estimation bias of $Z(s,a)$ is

$$\begin{aligned}
\mathbb{E}[Z(s,a)] &= \gamma\mathbb{E}\left[\min_a \max_{i\in\{1,2,\ldots,n\}} Q_i^{\text{est}}(s,a) - \min_{a'} Q^{\text{true}}(s,a')\right] \\
&= \gamma\mathbb{E}\left[\min_a \max_{i\in\{1,2,\ldots,n\}} E_i(s,a)\right] \\
&= \gamma\int_{-\varsigma}^{\varsigma} xmf_{\max}(x)[1 - F_{\max}(x)]^{m-1}\mathrm{d}x \\
&= \gamma\int_{-\varsigma}^{\varsigma} xmnf(x)[F(x)]^{n-1}\{1 - [F(x)]^n\}^{m-1}\mathrm{d}x \\
&= \gamma\int_{-\varsigma}^{\varsigma} \frac{mnx}{2\varsigma}\left(\frac{1}{2} + \frac{x}{2\varsigma}\right)^{n-1}\left\{1 - \left[\frac{1}{2} + \frac{x}{2\varsigma}\right]^n\right\}^{m-1}\mathrm{d}x \\
&= -\gamma\int_{-\varsigma}^{\varsigma} x\mathrm{d}\left\{1 - \left[\frac{1}{2} + \frac{x}{2\varsigma}\right]^n\right\}^m \\
&= -\gamma x\left\{1 - \left[\frac{1}{2} + \frac{x}{2\varsigma}\right]^n\right\}^m\Big|_{-\varsigma}^{\varsigma} + \gamma\int_{-\varsigma}^{\varsigma}\left\{1 - \left[\frac{1}{2} + \frac{x}{2\varsigma}\right]^n\right\}^m\mathrm{d}x
\end{aligned} \tag{11}$$

Let $w = \frac{1}{2} + \frac{x}{2\varsigma}$, and we have

$$\begin{aligned}
\mathbb{E}[Z(s,a)] &= -\gamma\varsigma + 2\gamma\varsigma\int_0^1 (1 - w^n)^m\mathrm{d}w \\
&= \gamma\varsigma\left[2\int_0^1 (1 - w^n)^m\mathrm{d}w - 1\right]
\end{aligned} \tag{12}$$

Substituting $t = w^n$ into Eq. (12), we have

$$\begin{aligned}
\mathbb{E}[Z(s,a)] &= \gamma\varsigma\left[\frac{2}{n}\int_0^1 t^{\frac{1}{n}-1}(1-t)^m\mathrm{d}t - 1\right] \\
&= \gamma\varsigma\left[\frac{2}{n}B\left(\frac{1}{n}, m+1\right) - 1\right] \\
&= \gamma\varsigma\left[\frac{2\Gamma(m+1)\Gamma\left(\frac{1}{n}\right)}{n\Gamma\left(m+\frac{1}{n}+1\right)} - 1\right]
\end{aligned} \tag{13}$$

To sum up, Theorem 2 is proved. ■

According to Theorem 2, we can draw the following conclusions:(1) When $n = 1$, it implies that only one Q network is used to estimate the Q value, and the expected estimation bias is $\gamma\varsigma\frac{1-m}{m+1} \leqslant 0$. This finding is consistent with the result in Theorem 1, thereby validating the proposed minmax operation. In this case, the system suffers from the underestimation problem.(2) When $n = 2$, it implies that two Q networks ($Q_1^{\text{est}}(s,a)$ and $Q_2^{\text{est}}(s,a)$) are used to estimate the Q value, and the expected estimation bias is $\mathbb{E}[Z(s,a)] = \gamma\varsigma\left[\frac{\Gamma(m+1)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(m+\frac{1}{2}+1\right)} - 1\right]$. We observe that when $m \leqslant 2$, the overestimation exists, i.e., $\mathbb{E}[Z(s,a)] > 0$. Conversely, when $m > 2$, the underestimation exists, i.e., $\mathbb{E}[Z(s,a)] < 0$. It is evident that our proposed minmax operation can control the estimation bias by changing the value of m.(3) Let $T = \frac{2\Gamma(m+1)\Gamma\left(\frac{1}{n}\right)}{n\Gamma\left(m+\frac{1}{n}+1\right)}$, and it is obvious that when $T = 1$, we have an unbiased estimation, i.e., $\mathbb{E}[Z(s,a)] = 0$. It can also be seen that our proposed minmax operation can balance underestimation and overestimation bias.(4) When $n \to \infty$, we can obtain $\mathbb{E}[Z(s,a)] = \gamma\varsigma$. This phenomenon indicates that, without changing m, the estimation bias tends to overestimate as n increases infinitely.

The overestimation and underestimation are undesirable, but they are not always be harmful.[40] The overestimation can cause the agent to incorrectly overestimate certain actions. This overestimation of some actions may lead to low returns, thereby making the agent's learning policy deviate from the optimal policy. However, the overestimation can help to encourage the agent to explore overestimated actions, which may discover some beneficial actions and help the agent to obtain the optimal policy. The underestimation can cause the agent to incorrectly underestimate certain actions, resulting in some potentially superior actions being ignored by the agent. In this case, the agent will easily fall into the local optimum, and reduce the algorithm performance in RL. In addition, although the underestimation hinders exploration to a certain extent, it can avoid useless exploration of the environment by the agent and improve the algorithm convergence. It can be seen that the overestimation and underestimation have benefits and drawbacks, but the performance of overestimation and underestimation in different environments cannot be guaranteed. Therefore, we need a method to control the estimation bias.

Our proposed minmax operation can effectively reduce the underestimation bias. Moreover, the conversion from underestimation to overestimation can be achieved, and the estimation bias can be controlled. Furthermore, the expected estimation bias with n and m is visualized through a heat map, as shown in Fig. 2. In order to better show these results, we set $\gamma\varsigma = 1$. The empirical results in Fig. 2 also support our discussion above. Under the condition of fixing m, by manipulating n, we can control the expected estimation bias and trend it towards 0. Similarly, we can tune m to control the expected estimation bias. Specifically, regardless of the environments, the value of m is determined by the given environment. In RL, we first construct the MDP model for the environment, including the action space. We can design the action space according to our requirements to determine the value of m. When m is determined, the value of n is determined based on the expression of expected estimation bias, so that the unbiased estimation is achieved or infinitely close to the unbiased

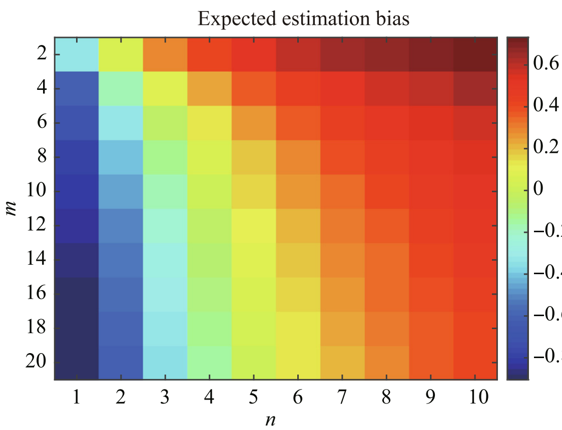

Fig. 2 The change of expected estimation bias with n and m.

estimation in a given environment. In conclusion, our proposed minmax operation can adapt to various environments.

4.3. Reduce the estimation variance

To demonstrate the superiority of our proposed minmax operation in reducing variance, we conduct a theoretical analysis as shown in Theorem 3.

Theorem 3. The variance of $\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)$ is

$$\mathrm{var}\left[\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)\right] = \frac{4n\varsigma^2}{(n+1)^2(n+2)} \tag{14}$$

Proof (Proof of Theorem 3).

For compactness, we rewrite $Q^{\max}(s,a)=\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)$ and $E^{\max}(s,a)=\max_{i\in\{1,2,\ldots,n\}} E_i(s,a)$. According to the definition of variance, it can be seen that $\mathrm{var}[Q^{\max}(s,a)]=\mathbb{E}\left\{[Q^{\max}(s,a)]^2\right\}-\mathbb{E}^2[Q^{\max}(s,a)]$. We can know that $Q^{\max}(s,a)=Q^{\mathrm{true}}(s,a)+E^{\max}(s,a)$, $Q^{\mathrm{true}}(s,a)$ is a constant, and $E^{\max}(s,a)$ is a random variable. Therefore, we have $\mathrm{var}[Q^{\max}(s,a)]=\mathbb{E}\left\{[E^{\max}(s,a)]^2\right\}-\mathbb{E}^2[E^{\max}(s,a)]$.

First, we calculate $\mathbb{E}[E^{\max}(s,a)]$. From the definition of variance, we can derive $\mathbb{E}[E^{\max}(s,a)]$ as

$$\begin{aligned}\mathbb{E}[E^{\max}(s,a)] &= \int_{-\varsigma}^{\varsigma} x f_{\max}(x)\mathrm{d}x\\ &= \int_{-\varsigma}^{\varsigma} xnf(x)[F(x)]^{n-1}\mathrm{d}x\\ &= \int_{-\varsigma}^{\varsigma} xn\tfrac{1}{2\varsigma}\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n-1}\mathrm{d}x\\ &= \int_{-\varsigma}^{\varsigma} x\mathrm{d}\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n}\\ &= x\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n}\Big|_{-\varsigma}^{\varsigma} - \int_{-\varsigma}^{\varsigma}\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n}\mathrm{d}x\\ &= \varsigma - \int_{-\varsigma}^{\varsigma}\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n}\mathrm{d}x\end{aligned} \tag{15}$$

Let $w=\frac{1}{2}+\frac{x}{2\varsigma}$, and Eq. (15) can be rewritten as

$$\begin{aligned}\mathbb{E}[E^{\max}(s,a)] &= \varsigma - \int_0^1 2\varsigma w^n\mathrm{d}w\\ &= \varsigma - \tfrac{2\varsigma}{n+1}w^{n+1}\Big|_0^1\\ &= \varsigma\tfrac{n-1}{n+1}\end{aligned} \tag{16}$$

Next, we calculate $\mathbb{E}\left\{[E^{\max}(s,a)]^2\right\}$, which is shown as

$$\begin{aligned}\mathbb{E}\left\{[E^{\max}(s,a)]^2\right\} &= \int_{-\varsigma}^{\varsigma} x^2 f_{\max}(x)\mathrm{d}x\\ &= \int_{-\varsigma}^{\varsigma} x^2 nf(x)[F(x)]^{n-1}\mathrm{d}x\\ &= \int_{-\varsigma}^{\varsigma} x^2 n\tfrac{1}{2\varsigma}\left(\tfrac{1}{2}+\tfrac{x}{2\varsigma}\right)^{n-1}\mathrm{d}x\end{aligned} \tag{17}$$

Let $w=\frac{1}{2}+\frac{x}{2\varsigma}$. Then, substituting it into Eq. (17), we have

$$\begin{aligned}\mathbb{E}\left\{[E^{\max}(s,a)]^2\right\} &= n\int_0^1 (2\varsigma w-\varsigma)^2 w^{n-1}\mathrm{d}w\\ &= n\varsigma^2\int_0^1 (2w-1)^2 w^{n-1}\mathrm{d}w\\ &= n\varsigma^2\int_0^1 (4w^2-4w+1)^2 w^{n-1}\mathrm{d}w\\ &= n\varsigma^2\int_0^1 (4w^{n+1}-4w^n+w^{n-1})\mathrm{d}w\\ &= n\varsigma^2\left(\tfrac{4}{n+2}w^{n+2}-\tfrac{4}{n+1}w^{n+1}+\tfrac{1}{n}w^n\right)\Big|_0^1\\ &= n\varsigma^2\left(\tfrac{4}{n+2}-\tfrac{4}{n+1}+\tfrac{1}{n}\right)\\ &= n\varsigma^2\tfrac{n^2-n+2}{n(n+1)(n+2)}\\ &= \tfrac{\varsigma^2\left(n^2-n+2\right)}{(n+1)(n+2)}\end{aligned} \tag{18}$$

Finally, we can get the following closed-form expression:

$$\begin{aligned}\mathrm{var}\left[\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)\right] &= \tfrac{\varsigma^2\left(n^2-n+2\right)}{(n+1)(n+2)} - \tfrac{\varsigma^2(n-1)^2}{(n+1)^2}\\ &= \tfrac{4n\varsigma^2}{(n+1)^2(n+2)}\end{aligned} \tag{19}$$

Therefore, Theorem 3 is proved. ■

According to Theorem 3, we can see that the variance of $\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)$ is only related to n. Specifically, when $n=1$, we can obtain $\mathrm{var}\left[\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)\right]=\frac{\varsigma^2}{3}$. When $n\to\infty$, we can obtain $\mathrm{var}\left[\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)\right]=0$. As the number of n increases, the variance decreases. This highlights how varying n can alter the estimation variance. To better visualize the relationship between $\mathrm{var}\left[\max_{i\in\{1,2,\ldots,n\}} Q_i^{\mathrm{est}}(s,a)\right]$ and n, we have plotted the variance variation for $n\in[1,2,\cdots,10]$ in Fig. 3. In order to better show the results, we set $\gamma\varsigma=1$. The results in Fig. 3 further support our discussion. The above analysis indicates that our proposed minmax operation can effectively control the variance.

4.4. Discussion

The theoretical derivation in Sections 4.2 and 4.3 demonstrates that our proposed minmax operation can effectively control estimation bias and reduce the variance. By controlling the estimation bias, the agent can estimate the Q values more accurately, thereby facilitating the agent to find the optimal policy. Similarly, the reduction of variance can enhance the network training stability, thereby promoting algorithms convergence. The above results theoretically demonstrate the superiority

Please cite this article in press as: HUANG F et al. Controlling underestimation bias in reinforcement learning via minmax operation, *Chin J Aeronaut* (2024), https://doi.org/10.1016/j.cja.2024.03.008

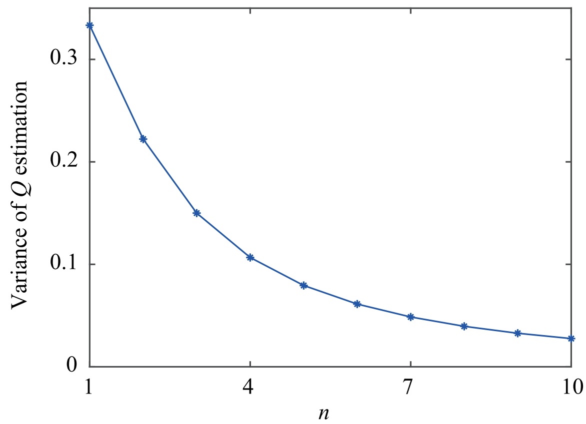

Fig. 3 Variance variation with n.

of the minmax operation. In Section 5, we empirically demonstrate the superiority of our proposed minmax operation through simulation experiments.

5. Experiments

In this section, the performance of our proposed minmax-DQN and minmax-DDQN algorithms is evaluated in three environments: (A) Lunarlander; (B) Catcher; (C) Pixelcopter. We also conduct comparative analysis with baseline algorithms DQN[12] and DDQN,[25] and state-of-the-art algorithms Averaged-DQN,[20] and Maxmin Q-learning.[27]

5.1. Environment setting

5.1.1. Lunarlander

Lunarlander is developed by OpenAI Gym to simulate the process of a spacecraft landing on the lunar surface,[41] as shown in Fig. 4. The goal for the purple spacecraft is to land safely on the flat ground between two yellow flags. The key player in this environment is the spacecraft, which acts as the agent and has four different actions at each step of the process: stop, fire the main engine, fire the left engine, and fire the right engine. The state space of the Lunarlander includes: the horizontal coordinate x, the vertical coordinate y, the horizontal velocity v_l, the vertical velocity v_v, the angle d, and the angular velocity v_d, whether leg 1 or leg 2 is in contact with the lunar surface. If the spacecraft crashes, the agent will receive a penalty of -100, whereas a successful landing between the yellow flags will result in a reward of $+100$. Furthermore, if the spacecraft maintains complete stationarity during landing, it will receive an additional reward of $+140$. In particular, each activation of the main engine costs -0.3. The spacecraft will receive a reward of $+10$ when leg 1 or leg 2 touches the ground.

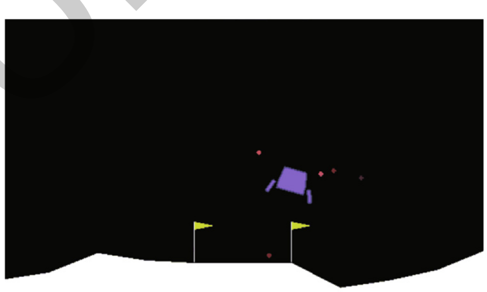

Fig. 4 Environment: Lunarlander.

5.1.2. Catcher

Catcher is a catching object game in the PyGame Learning Environment (PLE),[42] as shown in Fig. 5. In this game, the agent embodies as a white rectangle positioned at the bottom of the black playing area, and has two different actions: moving left and moving right. The red square randomly falls from the overhead, which can only move within the boundaries of the black area. The goal of the agent is to catch all the falling red squares. The agent receives a reward of $+1$ for successfully catching a red square, and a penalty of -1 for failure.

5.1.3. Pixelcopter

Pixelcopter is a side-scrolling game in the PLE,[42] as shown in Fig. 6. It is worth noting that Fig. 6 provides a portion of the game scene. The agent of the white square must successfully navigate a challenging cavern. If the agent touches the green area, the episode will end. The state space includes crucial variables such as position, speed, and distances to the upper and

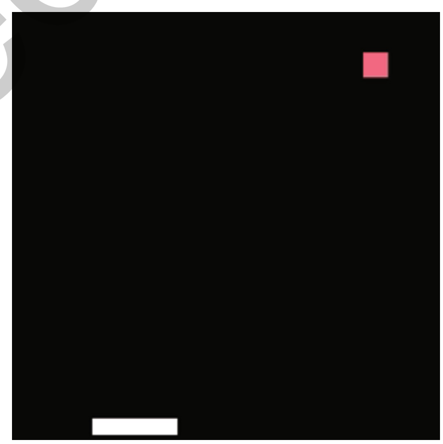

Fig. 5 Environment: Catcher.

Fig. 6 Environment: Pixelcopter.

Please cite this article in press as: HUANG F et al. Controlling underestimation bias in reinforcement learning via minmax operation, *Chin J Aeronaut* (2024), https://doi.org/10.1016/j.cja.2024.03.008

lower boundaries of the green area. For successfully passing each green block, the agent receives a reward of $+1$. However, any contact with the green area results in a penalty of -1. The goal of the agent is to quickly traverse the black areas while avoiding any contact with the treacherous green area.

5.2. Network structure and parameter settings

In the three environments Lunarlander, Catcher and Pixelcopter, we reuse the neural network architecture and fine-tune hyperparameters in Ref. [27]. Considering that the system state and action spaces are low-dimensional, all networks are set as a multi-layer perceptron neural network with two hidden layers. The number of nodes for two hidden layers is $[64, 32]$. In order to ensure fairness, the initialization of all network weight parameters is conducted in a uniform random initialization way. The networks input states and output Q values for each action. All experiments are completed on the server equipped with Intel CPU i9-9960X and NVIDIA Geforce 3090Ti GPU.

In parameters setting, the optimizer is RMSprop with gradient clip 5. The discount factor is set as 0.99. We set the experience replay memory of 20000 tuples, and update the Q network with a mini batch size of 64 at each step. The target network parameters update every 200 frames by replicating the Q network parameters. The learning rate and overall training steps are set differently for each environment. Specifically, for Lunarlander, Catcher, and Pixelcopter, the training steps are $1.5 \times 10^6, 3.5 \times 10^6$ and 2×10^6, respectively. In addition, the maximum number of steps per episode does not exceed 1000, and the algorithm performance is evaluated every 10 episodes. The ε-greedy exploration policy is adopted to achieve effective exploration to the environments. The exploration rate εlinearly decreases from 1.0 to 0.001 over 2000 steps, and is fixed to 0.001 after that. The parameter n is an variable value in our proposed minmax operation, and we mainly show the simulation results with $n = [4, 6]$. Each experiment is independently run 5 times, and the final simulation results are obtained by averaging the outcomes.

5.3. Simulation results

5.3.1. Comparison with published algorithms

To demonstrate the superiority of our proposed minmax-DQN and minmax-DDQN algorithms, we perform a comparative analysis against several algorithms, including DQN, DDQN, Averaged-DQN, and Maxmin Q-learning. For this evaluation, we set the parameter value n as 6. The simulation comparison curves for the three environments are shown in Figs. 7–9.

Figs. 7–9 show the average return as a function of training steps for different algorithms in various environments. The simulation results reveal that our proposed minmax-DQN and minmax-DDQN can achieve efficient network training, and significantly improve the performance of the baseline algorithms DQN and DDQN. Moreover, we can also see that our algorithms outperform Averaged-DQN and Maxmin Q-learning, mainly in terms of having the highest average return and the smallest fluctuation amplitude. Furthermore, it is worth noting that minmax-DQN and minmax-DDQN can achieve stable convergence and optimal performance with minimal training steps. In contrast, other algorithms require more

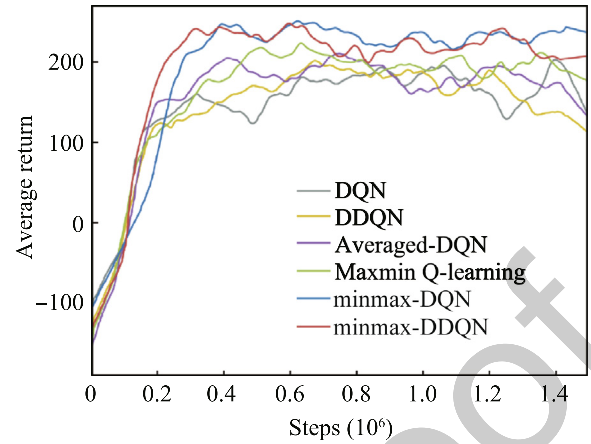

Fig. 7 Average return for different algorithms in Lunarlander environment.

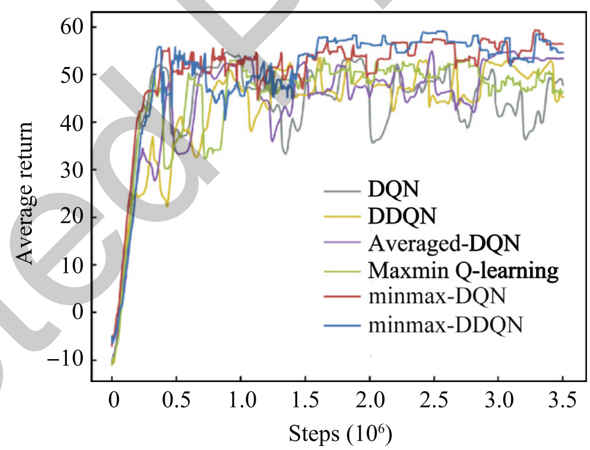

Fig. 8 Average return for different algorithms in Catcher environment.

training steps, and achieve sub-optimal performance with significant fluctuations. All these phenomena indicate the superiority of our proposed algorithms.

In addition, it can be seen that the performance of the algorithms varies between environments for two possible reasons. First, different environments perform differently under the underestimation and overestimation biases. Some environments may benefit from appropriate underestimation, while others may perform better with appropriate overestimation. Second, the hyperparameter n affects the performance of the algorithms. As shown in Figs. 7–9, the parameter n is fixed, but there is some variability in the performance of different environments with the same n. Therefore, in the following section, we discuss the effect of different values of n on the system performance for a comprehensive analysis.

5.3.2. Scalability

In the proposed minmax operation, the value of n is varying. In order to thoroughly understand the influence of n and demonstrate the robustness of our algorithms, we conduct a comprehensive performance analysis with $n = [4, 6]$. The main

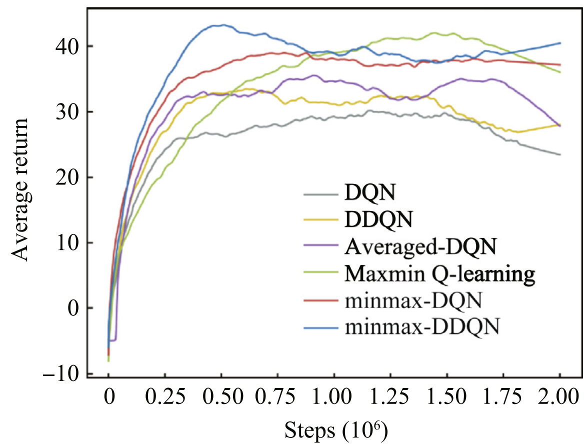

Fig. 9 Average return for different algorithms in Pixelcopter environment.

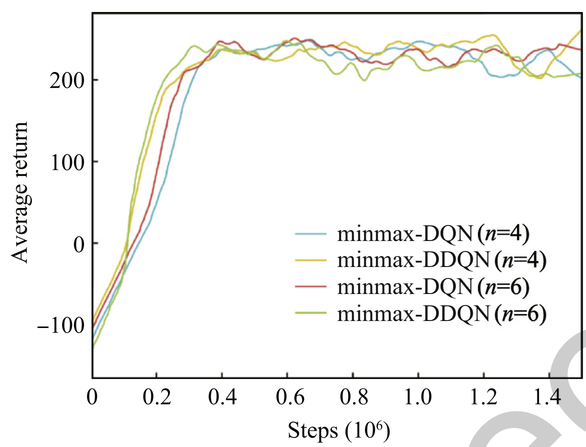

Fig. 10 Average return for proposed algorithms in Lunarlander environment with n.

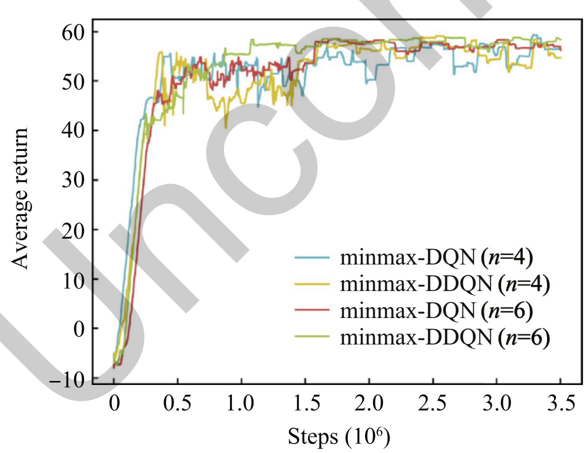

Fig. 11 Average return for proposed algorithms in Catcher environment with n.

results for the Lunarlander, Catcher, and Pixelcopter environments are shown in Figs. 10–12.

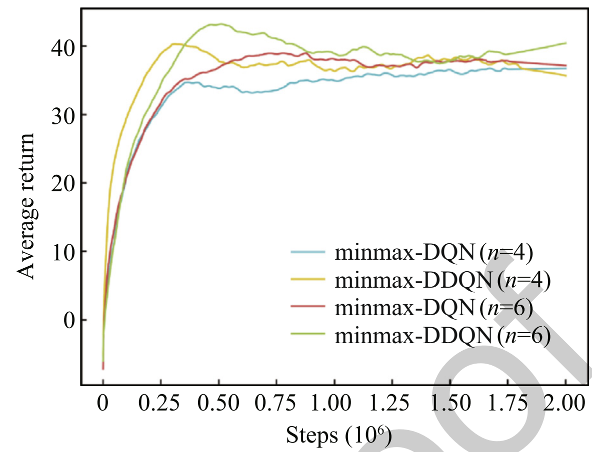

Fig. 12 Average return for proposed algorithms in Pixelcopter environment with n.

Figs. 10–12 show the functional relationship between the average return and training steps of our proposed algorithms under different n. It is obvious that when $n = [4, 6]$, the performance of our proposed algorithms remains stable and outperforms the baseline algorithms in all three environments. As n increases, the stability of the algorithms improves significantly, because the variance reduction helps to improve the stability of networks training. In general, when n is small, the algorithms learn fast. However, the performance is relatively weak. Conversely, when n is large, the algorithms learn slowly and perform better. This is because in the early stages, the agent attempts to explore the environment, which can help to achieve the desired performance. The above results fully illustrate the scalability and robustness of our proposed algorithms.

However, in the Lunarlander environment, there is a decrease in performance as n increases slightly. This can be attributed to the completion of the transition from underestimation to unbiased estimation to overestimation. In this process, different environments are affected differently by underestimation and overestimation, which cause small fluctuations in performance but do not affect the overall performance. These results provide valuable insight into performance trends and further highlight the effectiveness and robustness of our proposed minmax operation in different environments.

6. Conclusions

Overestimation and underestimation bias were two byproducts of Reinforcement Learning (RL), which could lead to inaccurate decisions in many fields, thereby affecting the system performance. While there were many studies that measured overestimation, few studies measured underestimation. To solve these problems, we conducted a theoretical analysis and measurement of the degree of underestimation bias by using the min operator. We proposed the minmax operation to flexibly control the estimation bias by adjusting the number of multiple state-action networks. Next, we designed two algorithms, named minmax-DQN and minmax-DDQN, which could improve the adaptability to the environments. Theoretically, we derived the closed-form expressions for the estimation bias and variance by using the minmax operation. The theoretical results showed that this operation could switch among overestimation, underestimation and unbiased estimation. Meanwhile, the minmax operation could significantly reduce the variance, and contribute to network training. Finally, vast experimental results indicated that our proposed minmax operation outperformed the baseline and state-of-the-art algorithms in various simulation environments. In addition, since our modification did not change the framework, it was highly applicable to other RL algorithms.

In the future, we plan to combine the minmax operation with other technologies for practical applications, such as the air target intention recognition and unmanned aerial vehicle combat.

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.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62173272).

References

[1]. Haddad TA, Aouag Hedjazi D. A deep reinforcement learning-based cooperative approach for multi-intersection traffic signal control. *Eng Appl Artif Intell* 2022;**114**:105019.

[2]. Zhang Q, Xie Z, Cao B, et al. A policy iteration method for improving robot assembly trajectory efficiency. *Chin J Aeronaut* 2023;**36**(3):436–48.

[3]. Jin Y, Wei S, Yuan J, et al. Hierarchical and stable multiagent reinforcement learning for cooperative navigation control. *IEEE Trans Neural Netw Learn Syst* 2023;**34**(1):90–103.

[4]. Wang S, Liu F, Chao T, et al. Robust spline-line energy management guidance algorithm with multiple constraints and uncertainties for solid rocket ascending. *Chin J Aeronaut* 2022;**35**(2):214–34.

[5]. He Y, Wang D, Huang F, et al. A V2I and V2V collaboration framework to support emergency communications in ABS-aided internet of vehicles. *IEEE Trans Green Commun Networking* 2023;**7**(4):2038–51.

[6]. Chen YT, Hu MH, Xu Y, et al. Locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks. *Chin J Aeronaut* 2023;**36**(4):338–53.

[7]. Zhang Q, Xie ZW, Cao BS, et al. A policy iteration method for improving robot assembly trajectory efficiency. *Chin J Aeronaut* 2023;**36**(3):436–48.

[8]. Ni WJ, Bi Y, Wu D, et al. Energy-optimal trajectory planning for solar-powered aircraft using soft actor-critic. *Chin J Aeronaut* 2022;**35**(10):337–53.

[9]. Ma B, Liu Z, Jiang F, et al. Reinforcement learning based UAV formation control in GPS-denied environment. Chin J Aeronaut 2023;Early Access, doi:10.1016/j.cja.2023.07.006.

[10]. Xi L, Zhang L, Xu YC, et al. Automatic generation control based on multiple-step greedy attribute and multiple level allocation strategy. *CSEE J Power Energy Syst* 2022;**8**(1):281–92.

[11]. Li T, Yang G, Chu J. Implicit posteriori parameter distribution optimization in reinforcement learning. IEEE Trans Cybern 2023;Early Access.

[12]. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;**518**(7540):529–33.

[13]. He Y, Wang D, Huang F, et al. A D2I and D2D collaboration framework for resource management in ABS-assisted post-disaster emergency networks. *IEEE Trans Veh Technol* 2024;**73**(2):2972–7.

[14]. Cheng R, Verma A, Orosz G, et al. Control regularization for reduced variance reinforcement learning. In: International conference on machine learning (ICML) 2019. p. 1–17.

[15]. Cicek DC, Duran E, Saglam B, et al. AWD3: Dynamic reduction of the estimation bias. In: International conference on tools with artificial intellignce (ICTAI) 2021. p. 775–9.

[16]. Fox R, Pakman A, Tishby N. Taming the noise in reinforcement learning via soft updates. arXiv preprint:1512.08562v4; 2017.

[17]. Lv J, Ma X, Yan J, et al. Efficient continuous control with double actors and regularized critics. In: Association for the Advancement of Artificial Intelligence (AAAI); 2022. p. 7655–63.

[18]. Hassam US, Marianno P, Ladislau B. Maximizing ensemble diversity in deep reinforcement learning. In: International conference on learning representations (ICLR); 2022. p. 1–25.

[19]. Gao A, Moon S, Kim J, et al. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. In: Conference and workshop on neural information processing systems (NeurIPS) 2021. p. 7436–47.

[20]. Anschel O, Baram N, Shimkin N. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In: International conference on machine learning (ICML); 2017. p. 176–85.

[21]. Lee K, Laskin M, Srinivas A, et al. SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning. In: International conference on machine learning (ICML); 2021. p. 1–20.

[22]. Cini A, D'Eramo C, Peters J, et al. Deep reinforcement learning with weighted Q-learning. arXiv preprint:2003.09280v3; 2020.

[23]. Song Z, Parr R, Carin L. Revisiting the softmax bellman operator: new benefits and new perspective. In: International conference on machine learning (ICML); 2019. p. 10368–83.

[24]. Hasselt H. Van. Double Q-learning. In: Conference and workshop on neural information processing systems (NeurIPS); 2010. p. 2613–21.

[25]. Hasselt H. Van, Guez A, Silver D. Deep reinforcement learning with double Q-learning. Association for the Advancement of Artificial Intelligence (AAAI) 2016. p. 2094–100.

[26]. Fujimoto S, Hoof H. van, Meger D. Addressing function approximation error in actor-critic methods. In: International conference on machine learning (ICML); 2018. p. 1582–91.

[27]. Lan Q, Pan Y, Fyshe A, et al. Maxmin Q-learning: Controlling the estimation bias of Q-learning. In: International conference on learning representations (ICLR); 2020. p. 1–19.

[28]. Li Z, Hou X. Mixing update Q-value for deep reinforcement learning. In: International joint conference on neural networks (IJCNN); 2019. p. 1–6.

[29]. Zhang Z, Pan z, Mykel J. Weighted double Q-learning. In: International joint conference on artificial intelligence (IJCAI); 2017. p. 3455–61.

[30]. Karimpanal TG, Le H, Abdolshah M, et al. Balanced Q-learning: Combining the influence of optimistic and pessimistic targets. arXiv preprint:2111.02787v1; 2021.

[31]. Wu D, Dong X, Shen J, et al. Reducing estimation bias via triplet-average deep deterministic policy gradient. *IEEE Trans Neural Netw Learn Syst* 2020;**33**(11):4933–45.

[32]. Sutton RS, Barto AG. *Reinforcement learning: An introduction*. Combridge: MIT Press; 2018.

[33]. Kang B, Zhao C. Deceptive evidence detection in information fusion of belief functions based on reinforcement learning. *Inf Fusion* 2024;**103**:102102.

[34]. Huang F, He Y, Deng X, et al. A novel discount-weighted average fusion method based on reinforcement learning for conflicting data. *IEEE Syst J* 2022;**17**(3):4748–51.
[35]. Yuan Y, Yu Z, Gu Z, et al. A novel multi-step q-learning method to improve data efficiency for deep reinforcement learning. *Knowl.-Based Syst* 2019;**175**:107–17.
[36]. Duan J, Guan Y, Li SE, et al. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Trans Neural Netw Learn Syst* 2022;**33**(11):6584–98.
[37]. Huang F, Deng X, He Y, et al. A novel policy based on action confidence limit to improve exploration efficiency in reinforcement learning. *Inf Sci* 2023;**640**:119011.
[38]. Thrun S, Schwartz A. Issues in using function approximation for reinforcement learning. Fourth Connectionist Models Summer School 1993. p. 25–263.
[39]. Karimpanal TG, Le H, Abdolshah M, et al. Balanced Q-learning: Combining the influence of optimistic and pessimistic targets. arXiv preprint:2111.02787v1; 2021.
[40]. Duan J, Guan Y, Ren Y, et al. Addressing value estimation errors in reinforcement learning with a state-action return distribution function. arXiv preprint:2001.02811v1; 2020.
[41]. Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym. arXiv preprint:1606.01540; 2016.
[42]. Tasfi N. PyGame learning environment [Internet]. Available from: https://github.com/ntas/PyGame-Learning-Environment.