Controlling Underestimation Bias in Reinforcement Learning via Quasi-Median Operation

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

School of Computer and Information Technology, Shanxi University, Taiyuan 030006, P.R. China
weiwei@sxu.edu.cn, 342564535@qq.com, ljy@sxu.edu.cn, lilynn1116@sxu.edu.cn, 202022407033@email.sxu.edu.cn

Abstract

How to get a good value estimation is one of the key problems in reinforcement learning (RL). Current off-policy methods, such as Maxmin Q-learning, TD3, and TADD, suffer from the underestimation problem when solving the overestimation problem. In this paper, we propose the Quasi-Median Operation, a novel way to mitigate the underestimation bias by selecting the quasi-median from multiple state-action values. Based on the quasi-median operation, we propose Quasi-Median Q-learning (QMQ) for the discrete action tasks and Quasi-Median Delayed Deep Deterministic Policy Gradient (QMD3) for the continuous action tasks. Theoretically, the underestimation bias of our method is improved while the estimation variance is significantly reduced compared to Maxmin Q-learning, TD3, and TADD. We conduct extensive experiments on the discrete and continuous action tasks, and results show that our method outperforms the state-of-the-art methods.

Introduction

As a widespread research problem in artificial intelligence, deep reinforcement learning has received increasing attention since its inception. Currently, deep reinforcement learning can solve many previously intractable problems, such as learning how to play video games directly from raw pixels (Mnih et al. 2013, 2015), autonomous navigation (Liu et al. 2021), gaming (Tesauro 1994; Baxter, Tridgell, and Weaver 2000; Silver et al. 2016, 2017; Vinyals et al. 2019), recommendation (Xiao and Wang 2021; Xie et al. 2021; Chen et al. 2018), and other fields.

The preliminary workload of deep reinforcement learning is significantly reduced by using powerful function approximators, such as multi-layer neural networks. However, the use of function approximators brings about estimation bias, which degrades the performance of reinforcement learning algorithms and hinders further extension of reinforcement learning to a broader range of domains. Overestimation is a common function approximation problem in reinforcement learning algorithms, such as Q-learning (Watkins and Dayan 1992) on the discrete action tasks and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2016) on the continuous control tasks. The overestimation problem is caused by inaccurate function estimation or maximization operation executed by Q-learning or its variants. Due to the positive bias resulting from overestimation, it is difficult for an agent to learn high-quality policy in many tasks (Thrun and Schwartz 1993; Szita and Lőrincz 2008; Strehl, Li, and Littman 2009).

Recently, some improved Q-learning methods have been proposed to alleviate the overestimation problem on discrete tasks. G-learning (Fox, Pakman, and Tishby 2015) regularizes the estimated values by penalizing deterministic policies at the beginning of the learning process. Averaged-DQN (Anschel, Baram, and Shimkin 2017) solves the overestimation problem by averaging the estimated values generated by multiple critics as the target state-action value to reduce the estimation error. Softmax Q-learning (Song, Parr, and Carin 2019) and Weighted Q-learning (D'Eramo, Restelli, and Nuara 2016) obtain more accurate target state-action value by weighting operation. The softmax operation and Gaussian approximation are adopted to weight multiple estimated values, respectively. However, for a limited number of action-value functions, the operations in these algorithms will never wholly eliminate the overestimation problem since the combination of several overestimation biases is always positive. Therefore, some methods address the overestimation problem by leveraging the decoupling or minimization operation on the discrete and continuous control settings.

Double Q-learning (van Hasselt 2010) and DDQN (van Hasselt, Guez, and Silver 2016) are two typical applications of the decoupling operation. They eliminate the overestimation problem by decoupling the two steps of selecting the greedy action and calculating the state-action value, respectively. Double Q-learning and DDQN solve the overestimation problem on the discrete action tasks, but they cannot be directly applied to the continuous control tasks. To solve this problem, Fujimoto et al. (Fujimoto, van Hoof, and Meger 2018) introduce a model-free reinforcement algorithm called Twin Delayed Deep Deterministic policy gradient (TD3), which successfully solves the overestimation problem in continuous control settings by taking the minimization operation on dual critics. Moreover, TD3 takes delayed policy update and target policy smoothing to reduce the error of each update. Similar to TD3, Maxmin Q-

*Corresponding author: Jiye Liang. Email: ljy@sxu.edu.cn.

learning (Lan et al. 2020) also uses the minimization operation to address the overestimation problem. Although Maxmin Q-learning and TD3 successfully solve the overestimation problem, they suffer from the underestimation problem. The underestimation can also degrade the performance of reinforcement learning algorithms (Wu et al. 2020). To address the underestimation problem, Wu et al. propose the Triplet-Average Deep Deterministic Policy Gradient (TADD) (Wu et al. 2020). In essence, TADD leverages a weighted approach to combine TD3 and Averaged-DQN. However, the weight of TADD is a fixed constant, which affects the performance of TADD when the settings of the experiment change. It is specifically noted that TADD still suffers from the underestimation problem.

In order to address the underestimation problem, in this paper, we propose the Quasi-Median Operation (QMO), which directly mitigates the underestimation bias by selecting the quasi-median value from multiple state-action values. In particular, the state-action value obtained by taking the quasi-median operation is not affected by outlier (e.g., extremely large or small) state-action values. Based on the quasi-median operation, we propose Quasi-Median Q-learning (QMQ) and Quasi-Median Delayed Deep Deterministic Policy Gradient (QMD3) to handle discrete and continuous action tasks, respectively. Theoretically, the absolute underestimation bias and the estimation variance of our method decrease monotonically with the number of critics increase. Moreover, the number of critics can balance the overestimation bias and the underestimation bias. In addition, we propose a trick called Exploration Improvement (EI) to improve the exploration ability of agent. We evaluate our algorithms on 8 MuJoCo tasks and 6 toy tasks wrapped via the OpenAI Gym API (Brockman et al. 2016) across a large number of seeds, and we perform ablation studies for each module. Extensive experiments demonstrate that our algorithm outperforms the state-of-the-art methods.

Preliminaries

The usual Reinforcement Learning framework can be formulated in terms of a Markov Decision Process (MDP), defined as a 6-tuple ($\mathcal{S}$, $\mathcal{A}$, ρ_0, p, r, γ). $\mathcal{S}$ and $\mathcal{A}$ denote the set of state and action spaces. p denotes transition distribution with conditional density function $p(s_{t+1}|s_t, a_t)$, along with ρ_0, denoting the initial density of the transition distribution. In addition, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function and outputs a scalar. $\gamma \in (0,1]$ denotes the discount factor.

At time t, given a state $s_t \in \mathcal{S}$ of the agent, the agent selects an action $a_t \in \mathcal{A}$ with respect to the conditional probability density $\pi_\phi(s_t|a_t)$ which modeled by a neural network with parameter ϕ. Then the agent receives a reward r and transfers to the next state s_{t+1}. The return is defined as the total discounted reward from time step t: $R_t = \sum_{l=t}^{T} \gamma^{l-t} r(s_l, a_l)$.

We now refer to the other concepts and notations utilized in the remainder of this paper. Finding the optimal policy π_ϕ that maximizes the expected return $J(\phi) = \mathbb{E}_{s_l \sim p_\pi, a_l \sim \pi}[R_0]$ is the ultimate goal of reinforcement learning. The state-action value is the expected return when performing action a in state s, and thereafter following π_ϕ: $Q^\pi(s, a) = \mathbb{E}_\pi[R_t]$. The Bellman equation provides a recursive relationship between the previous state and the subsequent state with a transition (s, a, s', a'):

$$Q^\pi(s, a) = r + \gamma \mathbb{E}_{s', a'}[Q^\pi(s', a')], a' \sim \pi(s'). \quad (1)$$

For the discrete algorithms, i.e., Maxmin Q-learning, the value function can be estimated with a differentiable function approximator such as a neural network (Mnih et al. 2015). The parameters of the critic approximator is θ, and θ is updated by minimizing the loss function:

$$L(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{B}}[(y - Q_\theta(s, a))^2], \quad (2)$$

where $y = r + \gamma \max_{a' \in \mathcal{A}} Q_\theta^{min}(s', a')$. Q_θ^{min} represents the critic obtained after taking the minimization operation on $Q_{\theta_i}(s, a)$ for $i = 1, \cdots, n$.

For the continuous control settings, i.e., DDPG, DDPG updates its actor parameter ϕ according to

$$\nabla_\phi J(\phi) = \mathbb{E}_{s \sim p_\pi}[\nabla_a Q_1^\pi(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)]. \quad (3)$$

θ is updated in the same way as Equation (2), with the different target value that $y^{DDPG} = r + \gamma Q'_{\theta'}(s', a'), a' \sim \pi_{\phi'}(s')$.

TD3 is a variant of DDPG, similar to the loss function of DDPG but with the target value that

$$y^{TD3} = r + \gamma \min_{i=1,2} Q'_{\theta'_i}(s', a'), a' \sim \pi_{\phi'}(s'), \quad (4)$$

where $\min_{i=1,2} Q'_{\theta_i}$ represents the minimization operation.

TADD is an extension of TD3, combining TD3 and Averaged-DQN. The target value y^{TADD} is defined as below:

$$y^{TADD} = r + \gamma(\beta \min_{i=1,2} Q'_{\theta'_i}(s', a') + (1 - \beta) Q'_2), \quad (5)$$

where $Q'_2 = \frac{1}{2}(Q'_{\theta'_3}(s', a') + Q'_{\theta'_4}(s', a'))$ denotes the average operation, and β is the weight of $\min_{i=1,2} Q'_{\theta'_i}(s', a')$ and Q'_2. In particular, the parameters of $Q'_{\theta'_3}$ at the previous moment are the same as $Q'_{\theta'_4}$.

The Proposed Method

In this section, we first develop the quasi-median operation, a simple and effective method to address the underestimation problem. Then, based on the quasi-median operation, we propose a novel variant of DQN called Quasi-Median Q-learning (QMQ) for the discrete action tasks and present a modified version of TD3 called Quasi-Median Delayed Deep Deterministic Policy Gradient (QMD3) for the continuous action tasks. In addition, we theoretically demonstrate the superiority of QMQ and QMD3 in improving underestimation bias and variance reduction.

Quasi-Median Operation

To obtain a more accurate target state-action value, one usually uses the average operation and the median operation, but these operations suffer from the overestimation problem (Wu et al. 2020). To cope with the overestimation problem,

Algorithm 1: QMQ algorithm

Initialize action-value functions $Q_{\theta_1}, \cdots, Q_{\theta_n}$
Initialize replay buffer $\mathcal{B}$
Observe initialize state s
while *interacting with the Environment* **do**
  $Q_\theta^{QMO}(s,a) \leftarrow \underset{i=1,\cdots,n}{\text{QMO}} Q_{\theta_i}(s,a)$
  Select action a by $\epsilon-$greedy based on $Q_\theta^{QMO}(s,a)$
  Take action a, observe reward r and new state s'
  Store transition tuple (s,a,r,s') in $\mathcal{B}$
  for $t = 1$ to T **do**
    Sample a mini-batch transitions (s,a,r,s') from $\mathcal{B}$
    Update target: $y \leftarrow r + \gamma \text{max}_{a' \in \mathcal{A}} Q_\theta^{QMO}(s',a')$
    Update Q_i: $Q_i(s,a) \leftarrow Q_i(s,a) + \alpha[y - Q_i(s,a)]$
  end for
  $s \leftarrow s'$
end while

one introduces the minimization operation (Lan et al. 2020; Fujimoto, van Hoof, and Meger 2018; Wu et al. 2020). However, this operation causes the underestimation problem. To find a trade-off between the overestimation and the underestimation, we propose a novel method called the quasi-median operation.

Let $Q_{(i)}$ denote the ith order statistic among n state-action values, and the quasi-median operation is defined as the selection of $Q_{(\lfloor \frac{n}{2} \rfloor)}(n > 3)$ from $Q_1, Q_2, \cdots, Q_n$. The quasi-median critic $Q_\theta^{QMO}(s,a)$ is defined as below:

$$Q_\theta^{QMO}(s,a) = \underset{i=1,\cdots,n}{\text{QMO}} Q_{\theta_i}(s,a),$$

where $\underset{i=1,\cdots,n}{\text{QMO}}$ represents the quasi-median operation taken on $Q_{\theta_i}(s,a)$ for $i = 1, 2, \cdots, n$.

The quasi-median operation can be used in any standard off-policy model-free reinforcement learning algorithm to alleviate the underestimation problem. To concretize the use of quasi-median operation, we design Quasi-Median Q-learning (QMQ) to handle discrete action tasks and Quasi-Median Delayed Deep Deterministic Policy Gradient (QMD3) to tackle continuous control tasks. The details about QMQ and QMD3 are shown in Algorithm 1 and Algorithm 2.

In particular, the parameters of the actor in QMD3 are updated as follows:

$$\bigtriangledown_\phi J(\phi) = \mathbb{E}_{s \sim p_\pi}[\bigtriangledown_a \frac{1}{n} \sum_{i=1}^{n} Q_{\theta_i}(s,a)|_{a=\pi_\phi(s)} \bigtriangledown_\phi \pi_\phi(s)].$$

where n represents the number of critics.

We use the mean state-action value from all critics to update the actor, while in TD3 and TADD, the actor updates its parameters with a fixed critic, such as the first critic Q_1. The multiple critics used by QMD3 can cover a wide range of state-action values in reasonable policy spaces. So we update the actor using multiple critics can encourage more effective exploration, and the actor with higher exploration ability, in turn, makes the critics more diverse and robust. We call this trick Exploration Improvement (EI).

In addition, we verify that our method can converge to the optimal policy in the finite MDP setting. Please refer to Appendix B for the proof. To better understand the differences among QMQ, QMD3, and other algorithms, we visualize several forward propagation processes of Maxmin Q-learning, QMQ, TADD, and QMD3 in Figure 1. Additional propagation processes figures for Q-learning, DDPG, and TD3 can be found in Appendix C.

Improving Underestimation Bias via Quasi-Median Operation

In this section, we analyze the underestimation phenomenon that occurs when taking the minimization operation. We illustrate its negative impact on Maxmin Q-learning and recent actor-critic algorithms, TD3, and TADD, which share the common feature of the minimization operation. We then demonstrate the advantage of QMQ and QMD3 in the estimation bias of state-action values.

We theoretically analyze the distribution of the ranked estimation bias by giving Lemma 1, which will help derive the estimation bias's expectation and variance after using the quasi-median operation.

Lemma 1. *Let $Q^{true}(s,a)$ be the true state-action value, and suppose that there are n independent $Q_i^{approx}(s,a)$ from different approximators for $i = 1, \cdots, n$. We define the estimation bias $Z_i(s,a)$ as $Z_i(s,a) = Q_i^{approx}(s,a) - Q^{true}(s,a)$. For $\forall$ $Z_i(s,a)$ with probability distribution function (pdf) $p(z)$ and cumulative distribution function (cdf) $F(z)$, the pdf of the kth order statistic $Z_{(k)}(s,a)$ is*

$$p_{(k)}(z) = \frac{n!}{(k-1)!(n-k)!}(F(z))^{k-1}(1-F(z))^{n-k}p(z).$$

For the proof please refer to Appendix A.A. Based on Lemma 1, we can deduce two important special cases:

$$p_{(1)}(z) = n(1-F(z))^{n-1}p(z), p_{(n)}(z) = n(F(z))^{n-1}p(z).$$

Considering $p_{(1)}(z)$ for $n = 2$, $p_{(1)}(z)$ represents the *pdf* for selecting the minimum from two values, which is similar to Maxmin Q-learning and TD3 for selecting target state-action value.

Subsequently, we analyze the expectation of the ordered bias $Z_{(k)}(s,a)$ by Theorem 1.

Theorem 1. *For $\mu \gg \lambda > 0$, we define the estimation bias $Z_i(s,a)$ which identically uniformly distributed in $[\lambda - \mu, \lambda + \mu]$. The expectation of $Z_{(k)}(s,a)$ is*

$$E[Z_{(k)}(s,a)] = \frac{2k-n-1}{n+1}\mu + \lambda, \tag{6}$$

where λ denotes the mean of the estimation bias, and μ represents half of the range of the estimation bias.

For the proof please refer to Appendix A.B.

According to Theorem 1, we can draw the following conclusions:

(1) The expectation of the estimation bias taking the minimization operation on two critics is $-\frac{1}{3}\mu + \lambda < 0$ when

Algorithm 2: QMD3 algorithm

Initialize critic networks $Q_{\theta_1}, \cdots, Q_{\theta_n}$ and initialize actor network π_ϕ with random parameters $\theta_1, \cdots, \theta_n, \phi$
Initialize target networks $\theta_1' \leftarrow \theta_1, \cdots, \theta_n' \leftarrow \theta_n, \phi' \leftarrow \phi$
Initialzie replay buffer $\mathcal{B}$
for $t = 1$ to T **do**
  Select action a with exploration noise $a \sim \pi_\phi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$, and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$
  Sample a mini-batch of B transitions (s, a, r, s') from $\mathcal{B}$
  $a' \leftarrow \pi_{\phi'}(s')$+clip$(\epsilon, -c, c), \epsilon \sim \mathcal{N}(0, \sigma)$
  $y \leftarrow r + \gamma\, Q_{\theta'}^{QMO}(s', \pi_{\phi'}(s'))$
  Update the critic θ_i by minimizing the Bellman loss: $B^{-1}\sum(y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update the actor ϕ by the deterministic policy gradient: $\bigtriangledown_\phi J(\phi) = B^{-1}\sum \bigtriangledown_a n^{-1} \sum Q_{\theta_i}(s, a)|_{a=\pi_\phi(s)} \bigtriangledown_\phi \pi_\phi(s)$
    Update target networks: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$, $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
  end if
end for

n and k are equal to 2 and 1, respectively. The estimation bias is identical to the expectation of the estimation bias in Maxmin Q-learning and TD3. Moreover, the expectation of the estimation bias of TADD is $-\frac{19}{60}\mu + 0.05\lambda < 0$. Thus, Maxmin Q-learning, TD3, and TADD suffer from the underestimation problem.

(2) The expectation of the estimation bias using the median operation on multiple critics is λ, which is the same as using the average operation, implying that both the median operation and the average operation cause the overestimation problem.

(3) The expectation of the estimation bias of QMQ and QMD3 is $-\frac{1}{n+1}\mu + \lambda$ after taking the quasi-median operation when n is an even number, and the expectation of the estimation bias of QMQ and QMD3 becomes $-\frac{2}{n+1}\mu + \lambda$ when n is an odd number.

Based on the above analysis, when $n > 3$, the expectation of the estimation bias of QMQ and QMD3 is greater than or equal to Maxmin Q-learning, TD3, and TADD. The quasi-median operation on the state-value functions degenerates to the minimization operation when $n \leq 3$. And as the number of critics increases, the expectation of the underestimation bias grows, which means our method can effective mitigate the underestimation problem. Note that using the quasi-median operation in QMQ and QMD3 can avoid the effect of extremely large or small state-action values.

In the following section, we analyze the variance of the estimation bias that can be effectively reduced by the quasi-median operation.

Reducing Estimation Variance via Quasi-Median Operation

To demonstrate the superiority of the quasi-median operation in variance reduction, we theoretically analyze the variance of estimation bias of Maxmin Q-learning, TD3, TADD, QMQ, and QMD3 in this section.

We analyze the variance of the order statistic $Z_{(k)}(s, a)$ by presenting Theorem 2, which can help us to derive the variance of the estimation bias of QMQ and QMD3.

Theorem 2. *The variance of $Z_{(k)}(s, a)$ is*

$$Var[Z_{(k)}(s, a)] = \frac{k(n-k+1)}{(n+1)^2(n+2)} 4\mu^2.$$

For the proof of Theorem 2, please refer to Appendix A.C.

According to Theorem 2, when $n = 2$ and $k = 1$, the variance of the estimation bias of TD3 and Maxmin Q-learning is $\frac{2}{9}\mu^2$. And, we can get the variance of the estimation bias $Var[Z_{(1)}(s, a)]$ is $\frac{1}{3}\mu^2$ when $n = 1$. Thus, the variance of the estimation bias of Averaged-DQN is $\frac{1}{3n}\mu^2$. Then the variance of the estimation bias of TADD is $0.201\mu^2$. In Corollary 1, we demonstrate the relationship between n and the variance of the estimation bias of QMQ and QMD3.

Corollary 1. *The variance of the estimation bias decreases as the number of critics, n, increases when $k = \lfloor \frac{n}{2} \rfloor$.*

For the proof, please refer to Appendix A.D.

We will illustrate the contribution of the different number of critics in the ablation experiment section. Considering the scenario where QMQ or QMD3 has the largest variance, i.e., $k = 2$ and $n = 4$, we thus calculate the variance of the estimation bias after taking the quasi-median operation on four critics. Then

$$Var[Z^{QMQ}(s, a)]|_{n=4}^{k=2} = Var[Z^{QMD3}(s, a)]|_{n=4}^{k=2} = \frac{4}{25}\mu^2.$$

Thus, we can find that the variance of the estimation bias of QMQ and QMD3 is smaller than other methods except for Averaged-DQN. However, Averaged-DQN yields an overestimation bias, which can seriously affect the performance. Furthermore, we give Corollary 2 to analyze the relationship between the variance of the estimation bias and the variance of the estimation, where the proof is in Appendix A.E.

Corollary 2. *The variance of the estimation bias is equivalent to the variance of the estimation, we have:*

$$Var[Z_{(k)}(s, a)] = Var[Q_{(k)}^{approx}(s, a)].$$

Based on the conclusions of Theorem 2 and Corollary 2, it is easy to know that the quasi-median operation can effectively reduce the variance of the estimation.

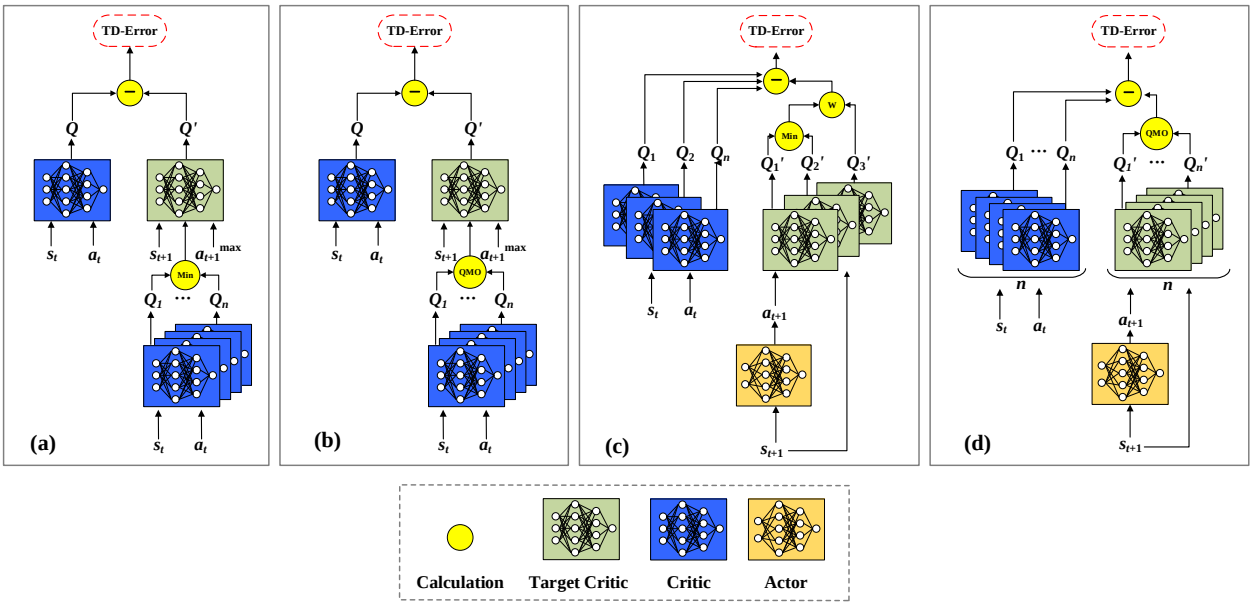
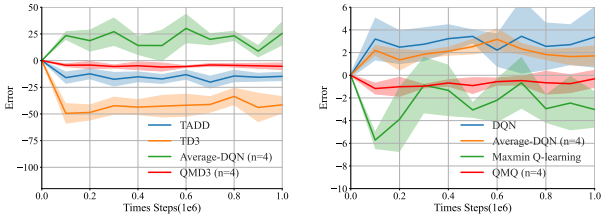


Figure 1: From left to right: (a) Maxmin Q-learning, (b) QMQ (ours), (c) TADD and (d) QMD3 (ours). TD-Error is obtained by subtraction (yellow and circled) between the current state-action value and the target state-action value, where "-" is the subtraction operation, "W" is the weighting operation, "Min" denotes the minimization operation, and "QMO" denotes the quasi-median operation.

In Figure 2, we measure the estimation bias of DQN, Maxmin Q-learning, Averaged-DQN, and QMQ on the discrete action task Space Invaders-v0, and we also measure the estimation bias of Averaged-DQN, TD3, TADD, and QMD3 on the continuous action task Hopper-v3. It is clear that our method can yield more accurate estimates and has a smaller estimation variance than other methods.



(a) Hopper-v3 (b) Space Invaders-v0

Figure 2: Measuring the estimation bias on Hopper-v3 (left) and Space Invaders-v0 (right). The darker lines represent the estimation bias of 10 random seeds. The shaded area represents one standard deviation.

Experiments

In this section, we empirically evaluate our method in the discrete and continuous action environments.

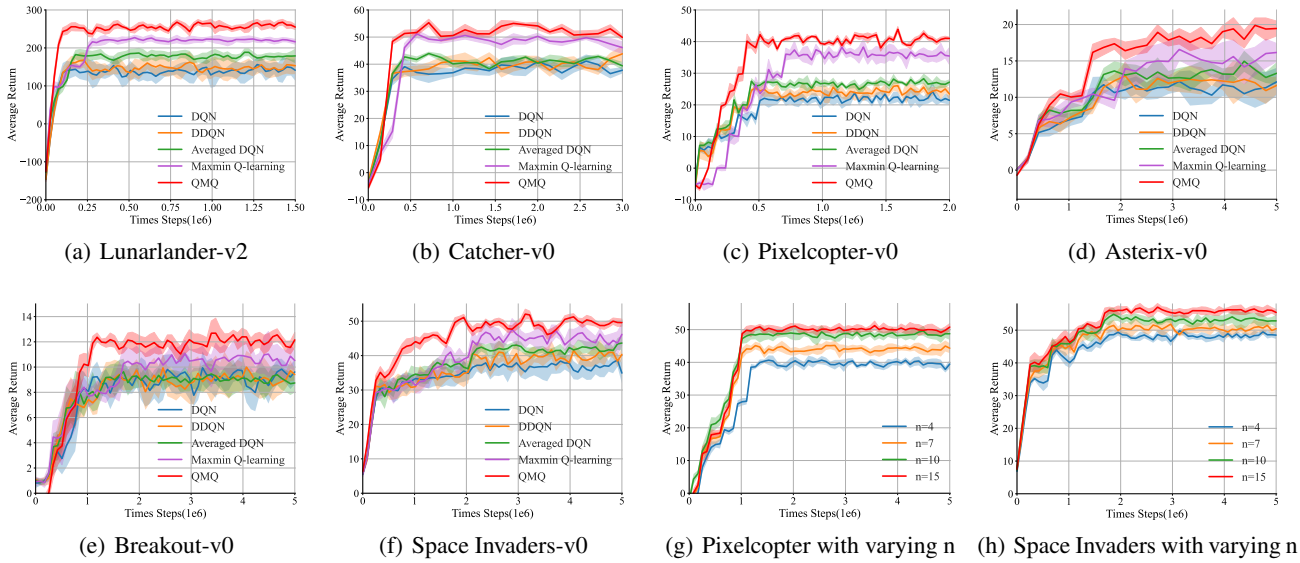
Discrete Action Environments

For the discrete action environments, we choose 6 games from Gym (Brockman et al. 2016), PLE (Tasfi 2016), and MinAtar (Young and Tian 2019): Lunarlander-v2, Catcher-v0, Pixelcopter-v0, Asterix-v0, Breakout-v0, and Space Invaders-v0 to evaluate QMQ. Moreover, we compare our QMQ with Maxmin Q-learning (Lan et al. 2020), DQN (Mnih et al. 2015), Double DQN (van Hasselt, Guez, and Silver 2016), and Averaged-DQN (Anschel, Baram, and Shimkin 2017).

For Lunarlander-v2, Catcher-v0, and Pixelcopter-v0, we reuse the hyper-parameters and settings of neural networks in (Lan et al. 2020). For MinAtar games (Asterix-v0, Breakout-v0, and Space Invaders-v0), the hyper-parameters and settings of neural networks are the same as those in (Young and Tian 2019). To ensure that our comparisons are fair and meaningful, we run our experiments on a large number of seeds with fair evaluation metrics. Considering the computational efficiency of training multiple critics, all discrete tasks, we set $n = 4$ for QMQ and Averaged-DQN, $n = 1$ for DQN, and $n = 2$ for Maxmin Q-learning. More detailed information about the rendering of the environment, hyper-parameters, and implementation details can be found in Appendix D.A and Appendix E.

Evaluation Figure 3 shows the training curves about the average return of each algorithm. From Figure 3, we can see that QMQ performs as well as or better than other algorithms on all tasks, while the stability of QMQ is better than those of other algorithms.

Ablation Studies We additionally test the performance of QMQ on Pixelcopter-v0 and Space Invaders-v0 with different n, as shown in Figure 3 (g) and (h). For larger n, QMQ learns faster and achieves better final performance



(a) Lunarlander-v2 (b) Catcher-v0 (c) Pixelcopter-v0 (d) Asterix-v0

(e) Breakout-v0 (f) Space Invaders-v0 (g) Pixelcopter with varying n (h) Space Invaders with varying n

Figure 3: Learning curves in the 6 benchmark environments. The curves are smoothed uniformly for better visualization. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Plots (h) and (i) show the performance of QMQ on Pixelcopter and Space Invaders with different n, and larger n leads to a better final performance in both environments.

with smaller estimation variance, which is consistent with the previous analysis.

Continuous Action Environments

For the continuous action environments, we compare the proposed QMD3 with DDPG (Lillicrap et al. 2016), TD3 (Fujimoto, van Hoof, and Meger 2018), Averaged-DQN (Anschel, Baram, and Shimkin 2017) and TADD (Wu et al. 2020) on 8 MuJoCo tasks (Todorov, Erez, and Tassa 2012): InvertedPendulum-v2 (IP), InvertedDoublePendulum-v2 (IDP), Reacher-v2, Hopper-v3, HalfCheetah-v3, Walker2d-v3, Ant-v3, and Humanoid-v3.

For all tasks, the hyper-parameters and settings of neural networks of QMD3 and Averaged-DQN are the same as those in TD3 (Fujimoto, van Hoof, and Meger 2018) and TADD (Wu et al. 2020). We set $n = 4$ for QMD3 and Averaged-DQN, $n = 1$ for DDPG, and $n = 2$ for TD3 and TADD. We run each algorithm 10 times to evaluate QMD3. For more details about the rendering of the environment, hyper-parameters, and implementation details, please refer to Appendix D.B and Appendix E.

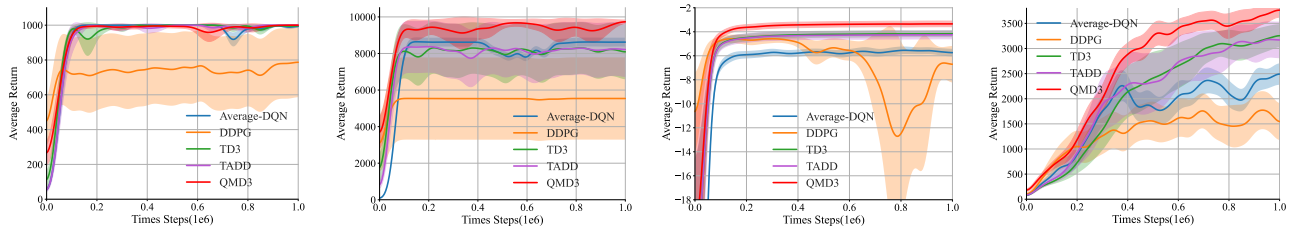
Evaluation Figure 4 shows that compared to DDPG, TD3, Averaged-DQN, and TADD, QMD3 can achieve better or comparable performance while they have similar convergence speeds on all continuous tasks. Especially for Walker2d-v3, Humanoid-v3, HalfCheetah-v3, and InvertedDoublePendulum-v2, QMD3 can achieve noticeably higher averaged return compared to TD3 and obtain the gains of 29%, 22.6%, 21.7% and 17.8%, respectively.

More detailed comparisons about max average return over 10 trials of 1 million time steps are shown in Table 1. From table 1, we can find that QMD3 exceeds all other algorithms in terms of the final performance, while DDPG and Averaged-DQN perform poorly on most tasks due to the overestimation problem. Moreover, we find that TD3 and TADD cannot work well for some tasks, such as Ant-v3. Especially, QMD3 owns superior stability to other comparative methods on most tasks except for Humanoid-v3, because every seed DDPG performs extremely worse on Humanoid-v3. Such a significant improvement is mainly attributed to the fact that QMD3 can effectively reduce the estimation error and enhance the exploration ability.

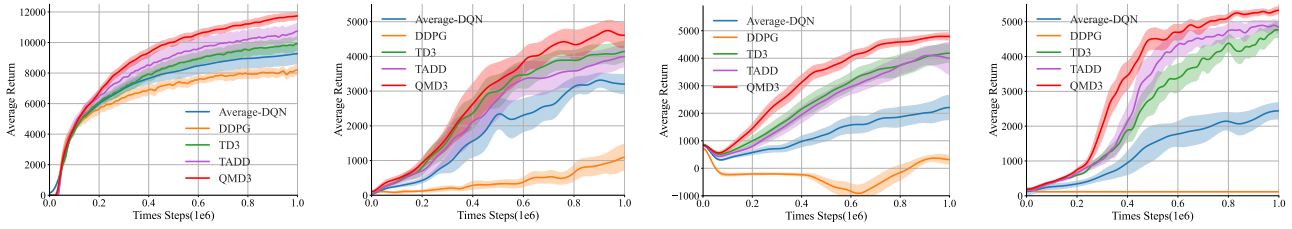
Table 1: Max Average return over 10 trials of 1 million time steps. The optimal value for each task, i.e., the maximum return and minimum variance, is bolded. $\pm$ corresponds to a single standard deviation.

Environment	QMD3	DDPG	TD3	TADD	Averaged-DQN
IP	**1000.0±0.0**	780.2	1000.0	1000.0	985.9
IDP	**9891.3±145.2**	5538.0	9237.4	9255.5	8621.7
Reacher	**-3.5±0.4**	-6.7	-4.0	-4.1	-5.7
Hopper	**3829.3±143.2**	1651.9	3464.7	3282.9	2562.3
HalfCheetah	**12221.9±387.5**	8070.4	9842.4	10605.7	9433.5
Walker2d	**4765.3±310.2**	1039.3	4449.3	4431.2	3625.5
Ant	**5034.2±315.3**	336.4	4229.0	4042.6	2751.9
Humanoid	**5332.4**±113.2	115.4	4719.8	4864.4	2440.5

Ablation Studies In Figure 5, we test the average return of QMD3 with different numbers of critics. The plots show that QMD3 is consistent with obtaining robust and superior performance with different numbers of critics. For larger n, QMD3 can achieve better final performance with smaller es-



(a) IP-v2 (b) IDP-v2 (c) Reacher-v2 (d) Hopper-v3

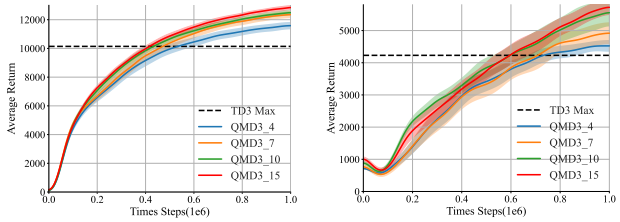


(e) HalfCheetah-v3 (f) Walker2d-v3 (g) Ant-v3 (h) Humanoid-v3

Figure 4: Learning curves on the 8 MuJoCo continuous control tasks. The curves are smoothed uniformly for better visualization. The darker lines represent the average evaluation of 10 random seeds. The shaded area represents half a standard deviation of average evaluation.

timation variance, consistent with Corollary 1.

To understand the contribution of the quasi-median operation and the exploration improvement, we compare the performance of removing each individual component from QMD3 and present the experimental results in Table 2. From Table 2, we can see that the quasi-median operation has a much more significant impact on the final performance than exploration improvement. These results suggest that the quasi-median operation plays a vital role in improving the performance of RL algorithms, but it should also be considered that training multiple critics increases time consumption. Moreover, exploration improvement also improves performance to some extent, so increasing the exploration ability of agent can be useful for RL performance improvement. In addition, only adding the quasi-median operation or exploration improvement leads to a clear improvement, but adding the combinations leads to higher performance.



(a) HalfCheetah with varying n (b) Ant with varying n

Figure 5: Learning curves on the 2 continuous control tasks with different n. The results are averaged over 10 runs, with the shaded area representing half a standard deviation. "TD3 Max" represents the optimal return of TD3.

Table 2: Average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over the quasi-median operation (QMO) and exploration improvement (EI). The maximum value for each task is bolded.

Method	HalfCheetah	Hopper	Walker2d	Ant
QMD3	**11763.5**	**3663.4**	**4424.5**	**4786.4**
TADD	10389.3	3145.7	3886.4	3933.4
TD3	9465.3	3351.7	4371.2	4155.6
Averaged-DQN	9213.6	2448.2	3265.9	2162.1
QMD3 - QMO	10576.3	3248.9	4103.6	4286.3
QMD3 - EI	11276.0	3472.8	4354.4	4534.2

Conclusion

Not only is overestimation considered a fundamental problem in reinforcement learning, but underestimation can also affect the performance of reinforcement learning algorithms. In this paper, we propose the quasi-median operation to address the underestimation problem. Furthermore, we apply the quasi-median operation to form Quasi-Median Q-learning for the discrete action tasks and Quasi-Median Delayed Deep Deterministic Policy Gradient for the continuous control tasks. From the theoretical point of view, our method's underestimation bias and stability are significantly improved compared to Maxmin Q-learning, TD3, and TADD algorithms. Extensive experimental results show that our proposed method significantly outperforms the state-of-the-art methods. Since our modifications are simple to implement, they can be easily used for any other value-based as well as actor-critic algorithms.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2020AAA0106100).

References

Anschel, O.; Baram, N.; and Shimkin, N. 2017. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. In *ICML*, 176–185.

Baxter, J.; Tridgell, A.; and Weaver, L. 2000. Learning to Play Chess Using Temporal Differences. *Machine Learning*, 40(3): 243–263.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR*, abs/1606.01540.

Chen, S.; Yu, Y.; Da, Q.; Tan, J.; Huang, H.; and Tang, H. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *KDD*, 1187–1196.

D'Eramo, C.; Restelli, M.; and Nuara, A. 2016. Estimating Maximum Expected Value through Gaussian Approximation. In *ICML*, 1032–1040.

Fox, R.; Pakman, A.; and Tishby, N. 2015. Taming the Noise in Reinforcement Learning via Soft Updates. *arXiv preprint arXiv:1512.08562*.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML*, 1582–1591.

Lan, Q.; Pan, Y.; Fyshe, A.; and White, M. 2020. Maxmin Q-learning: Controlling the Estimation Bias of Q-learning. In *ICLR*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous Control with Deep Reinforcement Learning. In *ICLR*.

Liu, M.; Zhao, F.; Niu, J.; and Liu, Y. 2021. ReinforcementDriving: Exploring Trajectories and Navigation for Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(2): 808–820.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature*, 518(7540): 529–533.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587): 484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T. P.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the Game of Go without Human Knowledge. *Nature*, 550(7676): 354–359.

Song, Z.; Parr, R.; and Carin, L. 2019. Revisiting the Softmax Bellman Operator: New Benefits and New Perspective. In *ICML*, 5916–5925.

Strehl, A. L.; Li, L.; and Littman, M. L. 2009. Reinforcement Learning in Finite MDPs: PAC Analysis. *Journal of Machine Learning Research*, 10(11): 2413–2444.

Szita, I.; and Lőrincz, A. 2008. The Many Faces of Optimism: A Unifying Approach. In *ICML*, 1048–1055.

Tasfi, N. 2016. PyGame Learning Environment. https://github.com/ntasfi/PyGame-Learning-Environment.

Tesauro, G. 1994. TD-Gammon, A Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, 6(2): 215–219.

Thrun, S.; and Schwartz, A. 1993. Issues in Using Function Approximation for Reinforcement Learning. In *Proceedings of the Fourth Connectionist Models Summer School*, 255–263.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A Physics Engine for Model-Based Control. In *IROS*, 5026–5033.

van Hasselt, H. 2010. Double Q-learning. In *NIPS*, 2613–2621.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, 2094–2100.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gülçehre, Ç.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T. P.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature*, 575(7782): 350–354.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine Learning*, 8(3-4): 279–292.

Wu, D.; Dong, X.; Shen, J.; and Hoi, S. C. H. 2020. Reducing Estimation Bias via Triplet-Average Deep Deterministic Policy Gradient. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11): 4933–4945.

Xiao, T.; and Wang, D. 2021. A General Offline Reinforcement Learning Framework for Interactive Recommendation. In *AAAI*, 4512–4520.

Xie, R.; Zhang, S.; Wang, R.; Xia, F.; and Lin, L. 2021. Hierarchical Reinforcement Learning for Integrated Recommendation. In *AAAI*, 4521–4528.

Young, K.; and Tian, T. 2019. Minatar: An Atari-Inspired Testbed for More Efficient Reinforcement Learning Experiments. *arXiv preprint arXiv:1903.03176, 2019*.