Mixing Update Q-value for Deep Reinforcement Learning

Zhunan Li[1,2], Xinwen Hou[1]
[1]Center for Research on Intelligent System and Engineering, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
lizhunan2017@ia.ac.cn, xwhou@nlpr.ia.ac.cn

***Abstract*—The value-based reinforcement learning methods are known to overestimate action values such as deep Q-learning, which could lead to suboptimal policies. This problem also persists in an actor-critic algorithm. In this paper, we propose a novel mechanism to minimize its effects on both the critic and the actor. Our mechanism builds on Double Q-learning, by mixing update action value based on the minimum and maximum between a pair of critics to limit the overestimation. We then propose a specific adaptation to the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) and show that the resulting algorithm not only reduces the observed overestimations, as hypothesized, but that this also leads to much better performance on several tasks.**

I. Introduction

Q-learning [1], [2] is one of the most popular reinforcement learning algorithms, but it is known to learn unrealistically high action value because it includes a maximization step over estimated action values, which tends to prefer overestimated to underestimated values [3]. The issue of value overestimation as a result of function approximation errors is well-studied in problems with discrete action spaces. And similar issues with actor-critic methods in continuous control tasks also attract more and more researchers' attention. The literature [4] shows that overestimation bias and accumulation of error in temporal difference methods are present in actor-critic settings, and also proposes a method to mitigate its impacts. Although the method can offer performance benefits, it also decreases training efficiency. Our proposed method not only addresses these issues, but also matches or outperforms the current state of the art.

Overestimation bias is a property of Q-learning in which the maximization of a noisy value estimate induces a consistent overestimation [4], [5]. This noise is unavoidable when given the inaccuracy of the estimator in function appropriation settings. So overestimations can occur because of the inaccurate action values. And this imprecision is further exacerbated by the nature of temporal difference learning [2], [6], in which an estimate of the value function is updated by bootstrapping. That means using an inaccurate estimate value to update will result in an accumulation of error, which can cause arbitrarily bad states to be estimated as higher value than their true value, leading to suboptimal policy. To address this concern, [3] proposes Double DQN algorithm which both yields more accurate value estimates and leads to much higher scores on several games.

At the same time, this overestimation property is also present for deterministic policy gradient in the continuous control setting [4]. In the discrete action setting, Double DQN is the ubiquitous solution to overestimation, but it is ineffective in an actor-critic setting [4]. Because Double DQN algorithm uses a independently trained critic to estimate the value of the current action with a separate target value function, and allows actions to be estimated without maximization bias. However, owing to the slowly varying policy in actor-critic settings, the current value and the target value estimations are too similar to avoid maximization bias. To address this issue, [4] proposes a clipped Double Q-learning variant which leverages the notion that a value estimate suffering from overestimation bias can be used as an approximate upper-bound to the true value estimate. This effectively minimizes the maximization bias. Because it takes the minimum value between a pair of critics and favors underestimations. As unlike overestimations, the underestimations will not be explicitly propagated through the policy update. In this paper, we extend this solution by proposing mixing update action value (Q-value), which is based on the minimum and maximum between a pair of critics with a mixing parameter. This methodology offers performance benefits for several games since it could balance the overestimation and underestimation.

Our adaptations are applied to the state of the art actor-critic method for continuous control, Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [4], to form the TD3 with mixing update Q-value algorithm. We evaluate our algorithm on eight continuous control domains from OpenAI gym [7], where we match or outperform the state of the art.

In the following sections, we will briefly introduce some basic concepts about reinforcement learning in section II, and then propose our methods in section III, after that we will evaluate our algorithm in section IV. Finally, we will discuss and conclude in section V and VI respectively.

II. Background

Reinforcement learning considers the paradigm of an agent interacting with an environment with the aim of maximizing the expected sum of discounted future rewards. At each timestep t, the agent takes an action a_t in the state s_t, and receives a scalar reward r_t and finds itself in a new state s_{t+1}. The return from a state is defined as the sum of discounted future rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $\gamma \in [0, 1]$ is a discount factor that trades off the importance of immediate and later rewards. Given a policy π, the true value of an action a_t in a state s_t is

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_i \sim p_{\pi}, a_i \sim \pi}[R_t | s_t, a_t], \tag{1}$$

which is known as the critic or the action value function. Note that p_{π} is the state-transition probabilities distribution for a policy π.

The goal in reinforcement learning is to learn the optimal policy π_{θ}, with parameters θ, which maximizes the expected return from the start distribution $J(\theta) = \mathbb{E}_{s_i \sim p_{\pi}, a_i \sim \pi}[R_0]$. For continuous control problem, the parametrized policies π_{θ} could be updated by taking the gradient of the expected return $\nabla_{\theta} J(\theta)$. In policy gradient methods, the policy, known as the actor in an actor-critic method, can be updated through the deterministic policy gradient algorithm [8]:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim p_{\pi}}[\nabla_a Q^{\pi}(s, a)|_{a=\pi(s)} \nabla_{\theta} \pi_{\theta}(s)]. \tag{2}$$

For a discrete and small state space, the value function $Q^{\pi}(s, a)$ can be learned using the tabular methods [2], such as Monte Carlo and temporal-difference methods, an update rule based on the Bellman equation [2]. The Bellman equation is a fundamental relationship between the value of a state-action pair (s_t, a_t) and the value of the subsequent state-action pair (s_{t+1}, a_{t+1}):

$$Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}}[Q^{\pi}(s_{t+1}, a_{t+1})]. \tag{3}$$

For a large state space, the value $Q^{\pi}(s, a)$ can be represented not as a table but as a differentiable function approximator $Q_w(s, a)$, with parameters w. In the DQN algorithm [9], [10], $Q_w(s, a)$ is the function computed by a multi-layer artificial neural network, with w the vector of connection weights in all the layers, and the network is updated by using Q-learning with a secondary frozen target network $Q_{w'}(s, a)$. The target network with parameters w', is the same as the online network except that its parameters are either periodically copied from the online network and kept fixed on all other steps, or by some proportion τ at each time step $w' \leftarrow \tau w + (1 - \tau) w'$. The target used by DQN is then

$$y_t = r_t + \gamma \max_{a_{t+1}} Q_{w'}(s_{t+1}, a_{t+1}), \tag{4}$$

where the actions a_{t+1} are selected from the policy $\pi(s_{t+1})$. This update could be applied in an off-policy approach, sampling random mini-batches of transitions from an experience replay buffer [4], [11].

However, the DQN algorithm involves max operator in the construction of its target policy, which makes it more likely to select overestimated values, resulting in overoptimistic value estimates. One way to view the problem is that it is due to using the same samples both to determine the maximizing action and to estimate its vlaue. A simple solution is to decouple the selection from the evaluation. Suppose we had two action-value functions, call them $Q_1(a)$ and $Q_2(a)$, each an estimate of the true value $q(a)$, for all actions a. We could then use one estimate, say $Q_1(a)$, to determine the maximizing action $a^* = \arg\max_a Q_1(a)$, and the other, $Q_2(a)$, to provide the estimate of its value, $Q_2(a^*) = Q_2(\arg\max_a Q_1(a))$. We can also repeat the process with the role of the two estimates reversed to yield a second estimate $Q_1(a^*) = Q_1(\arg\max_a Q_2(a))$. This is the idea behind Double DQN. And then the target of Double DQN can then be written as

$$y_t = r_t + \gamma Q_{w'}(s_{t+1}, \arg\max_{a_{t+1}} Q_w(s_{t+1}, a_{t+1})). \tag{5}$$

Notice that we select the greedy action according to the current values, as defined by w, but we use the second set of weights w' to fairly evaluate the value of this action.

III. Methods

The literature [4] theoretically analyzes overestimation bias in actor-critic methods and shows that this theoretical overestimation occur in practice for DDPG [12]. While some approaches to reducing overestimation bias have been proposed, they either are ineffective in an actor-critic setting, such as Double Q-learning, or can reduce overestimation in actor-critic settings but limited, such as clipped Double Q-learning. This section extends the clipped Double Q-learning by introducing a update mechanism for critics, which can improve the critic in actor-critic methods.

One major challenge of deep reinforcement learning is exploration. Like DDPG algorithm, we add the Ornstein-Uhlenbeck [13] noise, but clipped to the target policy. And then we find that it can offer performance benefits to the algorithm on several games. Because it may help the agent to better explore the environment.

A. Mixing Update Q-value for Critic

In Double Q-learning [2], [14], the greedy update is disentangled from the value function by maintaining two separate value estimates Q_{w_1} and Q_{w_2}, each of which is used to update the other. The idea of Double Q-learning is to reduce overestimations by decomposing the max operation in the target into action selection and action evaluation. Based on this idea, the authors of Double DQN propose using the target network as one of the value estimates, and obtain a policy by greedy maximization of the current value network rather than the target network. An analogous idea is also applied to the policy, using the current policy rather than the target policy to learn the target. In fact, however, due to the slow-changing policy in actor-critic approaches, the current network and taget network are too similar to make an independent estimation and offer little improvement. In addition, the critics are not entirely independent, because of the use of the oppsite critic in the learning targets, as well as the same replay buffer. To address these problems, the literature [4] uses a pair of critics (Q_{w_1}, Q_{w_2}) and actors $(\pi_{\theta_1}, \pi_{\theta_2})$, where π_{θ_1} is optimized with respect to Q_{w_1} and π_{θ_2} is optimized with respect to Q_{w_2} respectively, to simply upper-bound the less biased value estimate Q_{w_1} by the biased estimate Q_{w_2}. This result in taking the minimum between the two estimate, to give the target update of Clipped Double Q-learning:

$$y_t^1 = r_t + \gamma \min_{i=1,2} Q_{w_i'}(s_{t+1}, \pi_{\theta_1}(s_{t+1})), \quad (6)$$

$$y_t^2 = r_t + \gamma \min_{i=1,2} Q_{w_i'}(s_{t+1}, \pi_{\theta_2}(s_{t+1})). \quad (7)$$

Although the value target cannot introduce any additional overestimation over using the standard Q-learning target, this update rule may induce an underestimation bias. Since it always takes the minimum between the two estimated values. In order to achieve the same consequent of eliminating the overestimation and limit the effect of underestimation, we extend the solution and propose to mix the minimum and the maximum between the two estimates for updating target, rather than using exclusively the minimum. Mixing is accomplished using a positive parameter α with $\alpha < 1$. The overall mixed update target is expressed as follows:

$$Q(s_{t+1}, \pi_{\theta_1}(s_{t+1})) = \alpha \max_{i=1,2} Q_{w_i'}(s_{t+1}, \pi_{\theta_1}(s_{t+1})) + (1-\alpha) \min_{i=1,2} Q_{w_i'}(s_{t+1}, \pi_{\theta_1}(s_{t+1})), \quad (8)$$

$$y_t^1 = r_t + \gamma Q(s_{t+1}, \pi_{\theta_1}(s_{t+1})), \quad (9)$$

another target y_t^2 is similar to y_t^1. With the mixing parameter α, $Q(s_{t+1}, \pi_{\theta_1}(s_{t+1}))$ in (8) or (9) is not only always greater than the minimum and less than the maximum between the two value estimates, but also close to the minimum when α is close to zero. Especially when $\alpha = 0$, it degenerates into clipped Double Q-learning. Therefore, this value target will lead to a preference for Q-learning target with low-variance value estimates. In implementation, the costs can be decreased by using a single actor, and then we have the same target $y_t = y_t^1 = y_t^2$ for optimizing the parameters of Q_{w_1} and Q_{w_2}. We summarize an overview of this method in Algorithm 1.

B. *Clipped Ornstein-Uhlenbeck Noise for Actor*

A major challenge of learning in continuous action space is exploration. One advantage of off-policies algorithms is that they can better explore the action space for greater rewards, another advantage is that we can treat the problem of exploration independently from the reinforcement learning algorithm.

To help the actor explore the action space, similar to TD3 algorithm, we use a purely exploratory policy for the first one thousand or ten thousands time steps for the environment, after that we add Gaussian noise to each action whenever the actor interacts with the environment. When updating the critic network, we construct an exploration policy by adding clipped noise drawn from Ornstein-Uhlenbeck process to the target policy. This makes our modified target update:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{w_i'}(s_{t+1}, \pi_{\theta'}(s_{t+1}) + \epsilon), \quad \varepsilon \sim \text{clip}(\mathcal{N}_{ou}, -c, c) \quad (10)$$

where $\mathcal{N}_{ou}$ is Ornstein-Uhlenbeck process and the added noise is clipped to the target policy close to the original policy. Intuitively, it ensures that similar actions should have similar value.

Algorithm 1 TD3 with mixing update Q-value

```
Initialize critic networks $Q_{w_1}, Q_{w_2}$, and actor network $\pi_\theta$ with parameters $w_1, w_2, \theta$
Initialize target networks $w_1' \leftarrow w_1, w_2' \leftarrow w_2, \theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
for $t = 1$ to $T$ do
    Select action with exploration noise $a_t \leftarrow \pi_\theta(s_t) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward $r_t$ and new state $s_{t+1}$
    Store transition tuple $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
    Sample a random mini-batch of $N$ transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
    Set $\tilde{a} \leftarrow \pi_{\theta'}(s_{j+1}) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    Set $Q(s_{j+1}, \tilde{a}) \leftarrow \alpha \max_{i=1,2} Q_{w_i'}(s_{j+1}, \tilde{a}) + (1 - \alpha) \min_{i=1,2} Q_{w_i'}(s_{j+1}, \tilde{a})$
    Set $y_j \leftarrow r_j + \gamma Q(s_{j+1}, \tilde{a})$
    Update critic $w_i$ by minimizing the loss:
        $L(w_i) = \frac{1}{N} \sum_j (y_j - Q_{w_i}(s_j, a_j))^2$
    if $t$ mod $d$ then
        Update $\theta$ by the deterministic policy gradient:
            $\nabla_\theta J(\theta) = \frac{1}{N} \sum \nabla_a Q_{w_1}(s_t, a_t)|_{a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s_t)$
        Update target networks:
            $w_i' \leftarrow \tau w_i + (1-\tau) w_i'$
            $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
    end if
end for
```

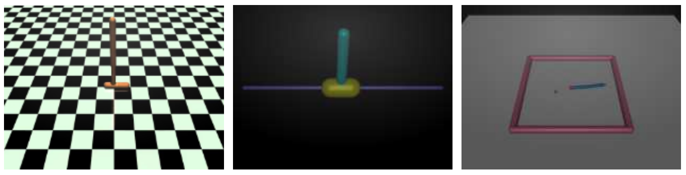

Fig. 1. Example MuJoCo environments. Hopper-v2(Left), InvertedPendulum-v2(Center), Reacher-v2(Right).

IV. Experiments

We improve the TD3 by applying the modification described in Section III-A and Section III-B respectively, to form two variant algorithms of TD3. In the next subsection, we will measure their performance for a selection of domains.

A. *Parameters and Evaluation*

The following parameters are used throughout the experiments. Similar to the TD3, both actor and critic employ 4-layers networks with 400, 300 nodes in the first and second hidden layer respectively, and use rectified linear units (ReLU) as the activation function between each layer, and a final tanh unit following the output of the actor. Both the actor and critic networks are also trained using the Adam Solver [15]. The discount factor γ is set to 0.99 and the critic

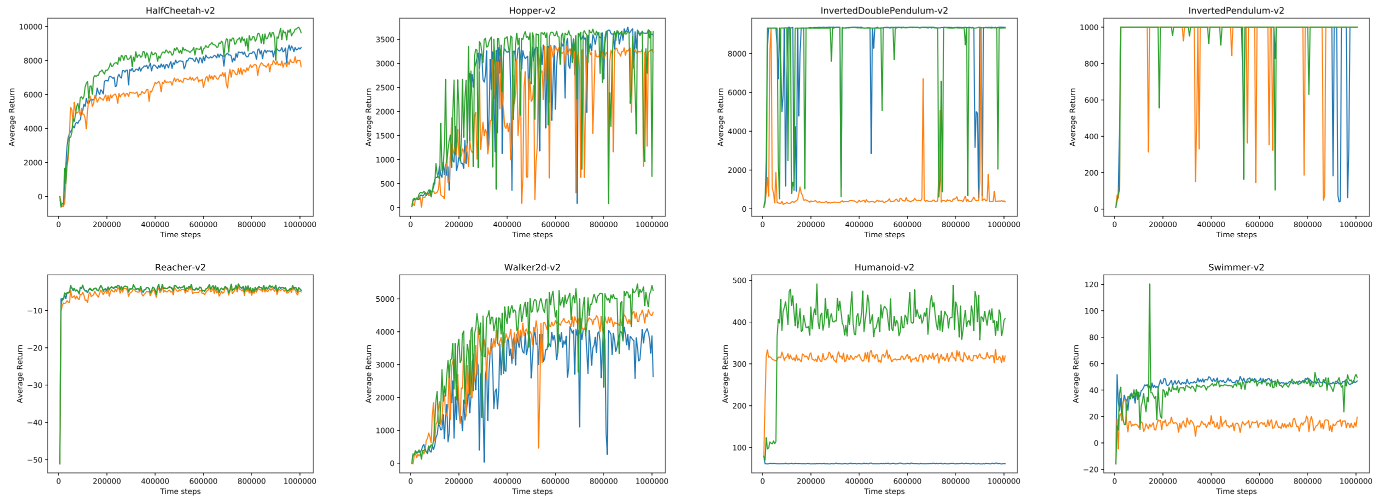

Fig. 2. Performance curves for the OpenAI gym continuous control task over 1 million tiem steps using variants of TD3 algorithm: original TD3 algorithm (navy blue), with Mixing Update Q-value (green), with Clipped Ornstein-Uhlenbeck Noise (orange). Mixing Update Q-value can offer performance benefits.

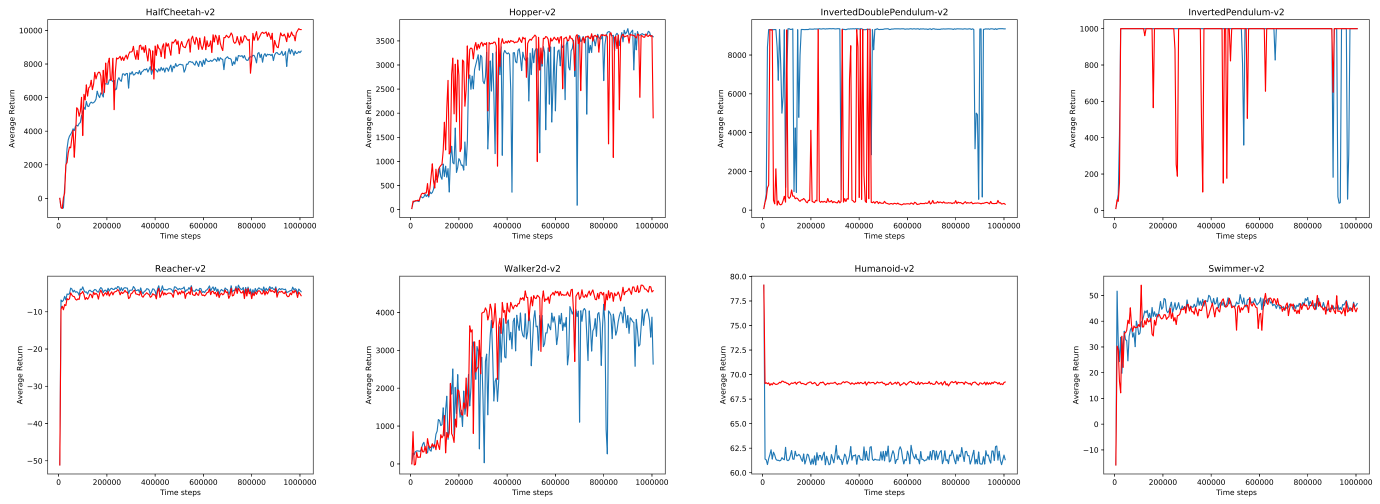

Fig. 3. Learning curves for the OpenAI gym continuous control task over 1 million tiem steps: original TD3 algorithm (navy blue), with Mixing Update Q-value and Clipped Ornstein-Uhlenbeck Noise (red). The resulting algorithm can match and even outperform the TD3 on some tasks.

TABLE I

AVERAGE RETURN OVER THE LAST 10 EPISODES OVER 1 MILLION TIME STEPS. MAXIMUM VLAUE FOR EACH TASK IS BOLDED.

Method	**HalfCheetah-v2**	**Hopper-v2**	**InvertedDoublePendulum-v2**	**Reacher-v2**	**Walker2d-v2**	**Humanoid-v2**	**Swimmer-v2**
TD3	8769.25	3586.25	**9350.72**	**-4.63**	2637.60	61.33	46.87
TD3+MUQ[a]	**9638.40**	**3664.30**	**9319.23**	**-4.52**	**5250.84**	**408.90**	**49.66**
TD3+COUN[b]	7636.99	3224.24	361.79	-5.01	4597.19	318.64	19.45
TD3+MQCN[c]	**10051.24**	1904.60	301.46	-5.69	4573.52	69.25	45.03

[a]The TD3 algorithm with mixing update Q-value (MUQ).
[b]The TD3 algorithm with clipped Ornstein-Uhlenbeck noise (COUN).
[c]The TD3 algorithm with mixing update Q-value and clipped Ornstein-Uhlenbeck noise (MQCN).

networks are trained with a mini-batch of a 100 transitions. The target networks, including actor and critic, are delayed updating every d iterations with $d = 2$. Both target networks are updated with $\tau = 0.005$, and the value of mixing parameter α is same as τ. With the mixing update Q-value, the target policy smoothing is implemented by adding $\epsilon \sim \mathcal{N}(0, 0.2)$ to the actions chosen by the target actor network, clipped to $(-0.5, 0.5)$. Otherwise, we replace the Gaussian noise with Ornstein-Uhlenbeck noise $\mathcal{N}_{ou}(\vartheta = 0.15, \mu = 0, \sigma = 0.2)$.

To evaluate our algorithm, we measure its performance on the suite of MuJoCo continuous control environment, interfaced through OpenAI Gym (Fig. 1) with no modifications to the environment or reward. In order to better explore and remove the dependency of parameters of the policy, we use a purely exploratory policy for the first ten thousands time steps on a environment (HalfCheetah-v2) and first one thousand time steps for the rest environments. The purely exploration policy is implemented by add Gaussian noise $\mathcal{N}(0, 0.1)$ to each action. Each task is run for 1 million time steps with evaluations every 5,000 time steps, where each evaluation reports the average return over 10 episodes with no exploration noise. Our results are reported with zero random seed of the Gym simulator and the network initialization.

We compare our two variants of TD3 against the original TD3 algorithm, the state of the art policy gradient algorithm. The comparison results and performance curves are in Fig. 2. It shows that the TD3 algorithm with mixing update Q-value matches or outperforms TD3 in both final performance and learning efficiency across most of tasks.

B. Fusion Study

We perform study to measure the performance of TD3 algorithm with both mixing update Q-value (Section III-A) and clipped Ornstein-Uhlenbeck noise (III-B). For each time step of this algorithm, the pair of critics are updated toward the mixing Q value selected by the target policy. We update the target policy $\pi_{\theta'}$ slowly $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$. And every d iterations, the policy π_θ is updated with respect to Q_{w_1} following the deterministic policy gradient algorithm. The overall mixed update target is expressed as follows:

$$Q(s_{t+1}, \pi_{\theta'}(s_{t+1}) + \epsilon) = \alpha \max_{i=1,2} Q_{w'_i}(s_{t+1}, \pi_{\theta'}(s_{t+1}) + \epsilon) + (1-\alpha) \min_{i=1,2} Q_{w'_i}(s_{t+1}, \pi_{\theta'}(s_{t+1}) + \epsilon), \tag{11}$$

$$y = r_t + \gamma Q(s_{t+1}, \pi_{\theta'}(s_{t+1}) + \epsilon), \quad \epsilon \sim \text{clip}(\mathcal{N}_{ou}, -c, c). \tag{12}$$

The learning curves in Fig. 3 in which we compare the performance of adding each improvement to TD3 without any modifications to the architecture and hyper-parameters. We also present our results in TABLE I in which we compare the performance of applying each different modifications to TD3 with the original TD3.

The importance of each improvement varies task to task. The combination of TD3 and mixing update Q-value can match or outperform the original TD3 algorithm across all tasks in not only final performance, but also learning efficiency. While the resulting TD3 with clipped Ornstein-Uhlenbeck noise outperforms TD3 algorithm by a wide margin on some tasks (e.g., Walker2d-v2 and Humanoid-v2), it performs at a much lower level in most environments. The TD3 algorithm with both of two improvements also outperforms the original TD3 on several tasks (e.g., HalfCheetah-v2 and Walker2d-v2). However, in the other cases, it cannot offer performance benefits and even it makes the performance of original algorithm worse. In summary, as the inclusion of mixing update Q-value into TD3 method matches or outperforms TD3, the state of the art algorithm, this suggests that mixing update Q-value is an effective measure to subdue the overestimations and limit the negative effect of underestimation.

V. Discussion

A. Why it could achieve better performance?

- As disscussed above, the clipped Double Q-learning is an effective way to eliminate the overestimation, but it also induces an underestimation bias. Although the underestimation is not propagated during learning, it may bring some negative impacts for performance. We introduce a mixing factor α that balances the overestimation and underestimation, it can reduce the overestimation while minimizing negative effects of underestimation.

B. Why do we choose the TD3 as the target algorithm for the modification, not DDPG or PPO?

- Obviously, the update rule for double Q-learning via a convex combination of the max and min values of two different action-value functions depends on a pair of critics. But neither PPO nor DDPG algorithm has two critics. In addition, the TD3 algorithm always takes the minimum between two estimates to update the target.

C. Does this method can be used for discrete action space?

- For discrete action settings, Double DQN is a better solution to reduce overestimation. It uses a single critic to estimate the value of action with a separate target critic. Compared with the algorithm that using two critics, the computational costs can be reduced. It would be valuable to see if the benefits of mixing update Q-value extend beyond the domain of continuous action space to other discrete action domains.

VI. Conclusion

Overestimation has been identified as a key problem in reinforcement learning methods. In this paper, we propose Mixing Update Q-value, a novel variant of Double Q-learning which can limit the effect of possible overestimation. Our results demonstrate that mitigating overestimation can improve the performance of reinforcement learning algorithm. This improvement is applied to the Twin Delayed Deep Deterministic policy gradient algorithm (TD3), which improves both the training efficiency and performance of TD3 in a number of challenging tasks in the continuous control settings. Our modification is simple to implement and improved algorithm matches or exceeds the performance of the state of the art algorithm.

Acknowledgment

Supported by the National Key R&D Program of China (No.2017YFC1200602) and the Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA19050203). Xinwen Hou is the corresponding author.

References

[1] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 2. Phoenix, AZ, 2016, pp. 2094–2100.

[4] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1587–1596. [Online]. Available: http://proceedings.mlr.press/v80/fujimoto18a.html

[5] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[6] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[11] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[14] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.