Action Candidate Based Clipped Double Q-learning for Discrete and Continuous Action Tasks

Haobo Jiang, Jin Xie[*], Jian Yang[*]
PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education
Jiangsu Key Lab of Image and Video Understanding for Social Security
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
{jiang.hao.bo, csjxie, csjyang}@njust.edu.cn

Abstract

Double Q-learning is a popular reinforcement learning algorithm in Markov decision process (MDP) problems. Clipped Double Q-learning, as an effective variant of Double Q-learning, employs the clipped double estimator to approximate the maximum expected action value. Due to the underestimation bias of the clipped double estimator, performance of clipped Double Q-learning may be degraded in some stochastic environments. In this paper, in order to reduce the underestimation bias, we propose an action candidate based clipped double estimator for Double Q-learning. Specifically, we first select a set of elite action candidates with the high action values from one set of estimators. Then, among these candidates, we choose the highest valued action from the other set of estimators. Finally, we use the maximum value in the second set of estimators to clip the action value of the chosen action in the first set of estimators and the clipped value is used for approximating the maximum expected action value. Theoretically, the underestimation bias in our clipped Double Q-learning decays monotonically as the number of the action candidates decreases. Moreover, the number of action candidates controls the trade-off between the overestimation and underestimation biases. In addition, we also extend our clipped Double Q-learning to continuous action tasks via approximating the elite continuous action candidates. We empirically verify that our algorithm can more accurately estimate the maximum expected action value on some toy environments and yield good performance on several benchmark problems.

Introduction

In recent years, reinforcement learning has achieved more and more attention. It aims to learn an optimal policy so that cumulative rewards can be maximized via trial-and-error in an unknown environment (Sutton and Barto 2018). Q-learning (Watkins and Dayan 1992) is one of widely studied reinforcement learning algorithms. As a model-free reinforcement learning algorithm, it generates the optimal policy via selecting the action which owns the largest estimated action value. In each update, Q-learning executes the maximization operation over action values for constructing the target value of Q-function. Unfortunately, this maximization operator tends to overestimate the action values. Due to the large positive bias, it is difficult to learn the high-quality policy for the Q-learning in many tasks (Thrun and Schwartz 1993; Szita and Lőrincz 2008; Strehl, Li, and Littman 2009). Moreover, such overestimation bias also exists in a variety of variants of Q-learning such as fitted Q-iteration (Strehl et al. 2006), delayed Q-learning (Ernst, Geurts, and Wehenkel 2005) and deep Q-network (DQN) (Mnih et al. 2015).

Recently, several improved Q-learning methods have been proposed to reduce the overestimation bias. Bias-corrected Q-learning (Lee, Defourny, and Powell 2013) adds a bias correction term on the target value so that the overestimation error can be reduced. Softmax Q-learning (Song, Parr, and Carin 2019) and Weighted Q-learning (D'Eramo, Restelli, and Nuara 2016) are proposed to soften the maximum operation via replacing it with the sum of the weighted action values. The softmax operation and Gaussian approximation are employed to generate the weights, respectively. In Averaged Q-learning (Anschel, Baram, and Shimkin 2017) and Maxmin Q-learning (Lan et al. 2020), their target values are constructed to reduce the bias and variance via combining multiple Q-functions.

Double Q-learning (Hasselt 2010; van Hasselt 2013; Zhang, Pan, and Kochenderfer 2017) is another popular method to avoid the overestimation bias. In Double Q-learning, it exploits the online collected experience sample to randomly update one of two Q-functions. In each update, the first Q-function selects the greedy action and the second Q-function evaluates its value. Although Double Q-learning can effectively relieve the overestimation bias in Q-learning in terms of the expected value, its target value may occasionally be with the large overestimation bias during training process. To avoid it, clipped Double Q-learning (Fujimoto, Van Hoof, and Meger 2018) directly uses the maximum action value of one Q-function to clip the target value of the Double Q-learning. Clipping Double Q-learning can be viewed as using the clipped double estimator to approximate the maximum expected value. However, the clipped double estimator suffers from the large underestimation bias.

In order to reduce the large negative bias of the clipped double estimator, in this paper, we propose an action candidate based clipped double estimator for Double Q-learning. Specifically, we first select a set of action candidates corresponding to high action values in one set of estimators. Then,

[*]Corresponding authors

among these action candidates, we choose the action with the highest value in the other set of estimators. At last, the corresponding action value of the selected action in the first set of estimators clipped by the maximum value in the second set of estimators is used to approximate the maximum expected value. Actually, in clipped Double Q-learning, the selected action from one Q-function is independent of the action evaluation in the other Q-function. Thus, the selected action may correspond to the low action value in the second Q-function, which results in the large underestimation. Through bridging the gap between the action selection and action evaluation from both Q-functions, our action candidate based clipped Double Q-learning can effectively reduce the underestimation bias. Theoretically, the underestimation bias in our clipped Double Q-learning decays monotonically as the number of action candidates decreases. Moreover, the number of action candidates can balance the overestimation bias in Q-learning and the underestimation bias in clipped Double Q-learning. Furthermore, we extend our action candidate based clipped Double Q-learning to the deep version. Also, based on the action candidate based clipped double estimator, we propose an effective variant of TD3 (Fujimoto, Van Hoof, and Meger 2018) for the continuous action tasks. Extensive experiments demonstrate that our algorithms can yield good performance on the benchmark problems.

Background

We model the reinforcement learning problem as an infinite-horizon discounted Markov Decision Process (MDP), which comprises a state space $\mathcal{S}$, a discrete action space $\mathcal{A}$, a state transition probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, an expected reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and a discount factor $\gamma \in [0, 1]$. At each step t, with a given state $s_t \in \mathcal{S}$, the agent receives a reward $r_t = R(s_t, a_t)$ and the new state $s_{t+1} \in \mathcal{S}$ after taking an action $a_t \in \mathcal{A}$. The goal of the agent is to find a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ that maximizes the expected return $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

In the MDP problem, the action value function $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$ denotes the expected return after doing the action a in the state s with the policy π. The optimal policy can be obtained as: $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$ where the optimal action value function $Q^*(s, a)$ satisfies the Bellman optimality equation:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{sa}^{s'} \max_{a' \in \mathcal{A}} Q^*(s', a'). \tag{1}$$

(Double) Q-learning To approximate $Q^*(s, a)$, Q-learning constructs a Q-function and updates it in each step via $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(y_t^Q - Q(s_t, a_t)\right)$, where the target value y_t^Q is defined as below:

$$y_t^Q = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'). \tag{2}$$

Instead, Double Q-learning maintains two Q-functions, Q^A and Q^B, and randomly updates one Q-function, such as Q^A, with the target value y_t^{DQ} as below:

$$y_t^{DQ} = r_t + \gamma Q^B \left(s_{t+1}, \arg\max_{a' \in \mathcal{A}} Q^A(s_{t+1}, a')\right). \tag{3}$$

Clipped Double Q-learning It uses the maximum action value of one Q-function to clip the target value in Double Q-learning as below to update the Q-function:

$$y_t^{CDQ} = r_t + \gamma \min\left\{Q^A(s_{t+1}, a^*), Q^B(s_{t+1}, a^*\right\}, \tag{4}$$

where $a^* = \arg\max_a Q^A(s_{t+1}, a)$.

Twin Delayed Deep Deterministic policy gradient (TD3) TD3 applies the clipped Double Q-learning into the continuous action control with the actor-critic framework. Specifically, it maintains a actor network $\mu(s; \boldsymbol{\phi})$ and two critic networks $Q(s, a; \boldsymbol{\theta}_1)$ and $Q(s, a; \boldsymbol{\theta}_2)$. Two critic networks are updated via $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \alpha \nabla_{\boldsymbol{\theta}_i} \mathbb{E}\left[\left(Q(s_t, a_t; \boldsymbol{\theta}_i) - y_t^{TD3}\right)^2\right]$. The target value y_t^{TD3} is defined as below:

$$y_t^{TD3} = r_t + \gamma \min_{i=1,2} Q\left(s_{t+1}, \mu\left(s_{t+1}; \boldsymbol{\phi}^-\right); \boldsymbol{\theta}_i^-\right) \tag{5}$$

where $\boldsymbol{\phi}^-$ and $\boldsymbol{\theta}_i^-$ are the soft updated parameters of $\boldsymbol{\phi}$ and $\boldsymbol{\theta}_i$. The actor $\mu(s; \boldsymbol{\phi})$ is updated via $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} J$, where the policy graident $\nabla_{\boldsymbol{\phi}} J$ is:

$$\nabla_{\boldsymbol{\phi}} J = \mathbb{E}\left[\nabla_a Q(s_t, a; \boldsymbol{\theta}_1)|_{a=\mu(s_t;\boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}} \mu(s_t; \boldsymbol{\phi})\right]. \tag{6}$$

Estimating the Maximum Expected Value

Revisiting the Clipped Double Estimator

Suppose that there is a finite set of N $(N \geq 2)$ independent random variables $\boldsymbol{X} = \{X_1, \ldots, X_N\}$ with the expected values $\boldsymbol{\mu} = \{\mu_1, \mu_2, \ldots, \mu_N\}$. We consider the problem of approximating the maximum expected value of the variables in such a set: $\mu^* = \max_i \mu_i = \max_i \mathbb{E}[X_i]$. The clipped double estimator (Fujimoto, Van Hoof, and Meger 2018) denoted as $\hat{\mu}_{CDE}^*$ is an effective estimator to estimate the maximum expected value.

Specifically, let $S = \bigcup_{i=1}^N S_i$ denote a set of samples, where S_i is the subset containing samples for the variable X_i. We assume that the samples in S_i are independent and identically distributed (i.i.d). Then, we can obtain a set of the unbiased estimators $\hat{\boldsymbol{\mu}} = \{\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_N\}$ where each element $\hat{\mu}_i$ is a unbiased estimator of $\mathbb{E}[X_i]$ and can be obtained by calculating the sample average: $\mathbb{E}[X_i] \approx \hat{\mu}_i \overset{\text{def}}{=} \frac{1}{|S_i|} \sum_{s \in S_i} s$. Further, we randomly divide the set of samples S into two subsets: S^A and S^B. Analogously, two sets of unbiased estimators $\hat{\boldsymbol{\mu}}^A = \left\{\hat{\mu}_1^A, \hat{\mu}_2^A, \ldots, \hat{\mu}_N^A\right\}$ and $\hat{\boldsymbol{\mu}}^B = \left\{\hat{\mu}_1^B, \hat{\mu}_2^B, \ldots, \hat{\mu}_N^B\right\}$ can be obtained by sample average: $\hat{\mu}_i^A = \frac{1}{|S_i^A|} \sum_{s \in S_i^A} s$, $\hat{\mu}_i^B = \frac{1}{|S_i^B|} \sum_{s \in S_i^B} s$. Finally, the clipped double estimator combines $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\mu}}^A$ and $\hat{\boldsymbol{\mu}}^B$ to construct the following estimator to approximate the maximum expected value:

$$\mu^* = \max_i \mu_i \approx \min\left\{\hat{\mu}_{a^*}^B, \max_i \hat{\mu}_i\right\}, \tag{7}$$

where the variable $\max_i \hat{\mu}_i$ is called the single estimator denoted as $\hat{\mu}_{SE}^*$ and the variable $\hat{\mu}_{a^*}^B$ is called the double estimator denoted as $\hat{\mu}_{DE}^*$.

Algorithm 1 Action Candidate Based Clipped Double Q-learning

```
Initialize Q-functions $Q^A$ and $Q^B$, initial state $s$ and the number $K$ of action candiadte.
1: repeat
2:    Select action $a$ based on $Q^A(s,\cdot)$, $Q^B(s,\cdot)$ (e.g., $\epsilon$-greedy in $Q^A(s,\cdot)+Q^B(s,\cdot)$) and observe reward $r$, next state $s'$.
3:    if update $Q^A$ then
4:       Determine action candidates $\boldsymbol{\mathcal{M}}_K$ from $Q^B(s',\cdot)$ and define $a_K^* = \arg\max_{a\in\boldsymbol{\mathcal{M}}_K} Q^A(s',a)$.
5:       $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\cdot\left(r+\gamma\min\left\{Q^B(s',a_K^*), \max_a Q^A(s',a)\right\} - Q^A(s,a)\right)$.
6:    else if update $Q^B$ then
7:       Determine action candidates $\boldsymbol{\mathcal{M}}_K$ from $Q^A(s',\cdot)$ and define $a_K^* = \arg\max_{a\in\boldsymbol{\mathcal{M}}_K} Q^B(s',a)$.
8:       $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)\cdot\left(r+\gamma\min\left\{Q^A(s',a_K^*), \max_a Q^B(s',a)\right\} - Q^B(s,a)\right)$.
9:    end if
10:   $s \leftarrow s'$
11: until end
```

For single estimator, it directly uses the maximum value in $\hat{\boldsymbol{\mu}}$ to approximate the maximum expected value. Since the expected value of the single estimator is no less than μ^*, the single estimator has overestimation bias. Instead, for double estimator, it first calculates the index a^* corresponding to the maximum value in $\hat{\boldsymbol{\mu}}^A$, that is $\hat{\mu}^A_{a^*} = \max_i \hat{\mu}^A_i$, and then uses the value $\hat{\mu}^B_{a^*}$ to estimate the maximum expected value. Due to the expected value of double estimator is no more than μ^*, it is underestimated.

Although the double estimator is underestimated in terms of the expected value, it still can't entirely eliminate the overestimation (Fujimoto, Van Hoof, and Meger 2018). By clipping the double estimator via single estimator, the clipped double estimator can effectively relieve it. However, due to the expected value of $\min\left\{\hat{\mu}^B_{a^*}, \max_i \hat{\mu}_i\right\}$ is no more than that of $\hat{\mu}^B_{a^*}$, the clipped double estimator may further exacerbate the underestimation bias in the double estimator and thus suffer from larger underestimation bias.

Action Candidate Based Clipped Double Estimator

Double estimator is essentially an underestimated estimator, leading to the underestimation bias. The clipping operation in the clipped double estimator further exacerbates the underestimation problem. Therefore, although the clipped double estimator can effectively avoid the positive bias, it generates the large negative bias.

In order to reduce the negative bias of the clipped double estimator, we propose an action candidate based clipped double estimator denoted as $\hat{\mu}^*_{AC}$. Notably, the double estimator chooses the index a^* only from the estimator set $\hat{\boldsymbol{\mu}}^A$ and ignores the other estimator set $\hat{\boldsymbol{\mu}}^B$. Thus, it may choose the index a^* associated with the low value in $\hat{\boldsymbol{\mu}}^B$ and generate the small estimation $\hat{\mu}^B_{a^*}$, leading to the large negative bias. Different from the double estimator, instead of selecting the index a^* from $\hat{\boldsymbol{\mu}}^A$ among all indexes, we just choose it from an index subset called candidates. The set of candidates, denoted as $\boldsymbol{\mathcal{M}}_K$, is defined as the index subset corresponding to the largest K values in $\hat{\boldsymbol{\mu}}^B$, that is:

$$\boldsymbol{\mathcal{M}}_K = \left\{ i \middle| \hat{\mu}^B_i \in \text{ top } K \text{ values in } \hat{\boldsymbol{\mu}}^{\mathrm{B}} \right\}. \tag{8}$$

The variable a_K^* is then selected as the index to maximize $\hat{\boldsymbol{\mu}}^A$ among the index subset $\boldsymbol{\mathcal{M}}_K$: $\hat{\mu}^A_{a_K^*} = \max_{i\in\boldsymbol{\mathcal{M}}_K} \hat{\mu}^A_i$. If there are multiple indexes owning the maximum value, we randomly pick one. Finally, by clipping, we estimate the maximum expected value as below:

$$\mu^* = \max_i \mu_i = \max_i \mathbb{E}\left[\hat{\mu}^B_i\right] \approx \min\left\{\hat{\mu}^B_{a_K^*}, \hat{\mu}^*_{SE}\right\}. \tag{9}$$

Consequently, we theoretically analyze the estimation bias of action candidate based clipped double estimator.

Theorem 1. *As the number K decreases, the underestimation decays monotonically, that is $\mathbb{E}\left[\min\left\{\hat{\mu}^B_{a_K^*}, \hat{\mu}^*_{SE}\right\}\right] \geq \mathbb{E}\left[\min\left\{\hat{\mu}^B_{a_{K+1}^*}, \hat{\mu}^*_{SE}\right\}\right]$, $1 \leq K < N$, where the inequality is strict if and only if $P\left(\hat{\mu}^*_{SE} > \hat{\mu}^B_{a_K^*} > \hat{\mu}^B_{a_{K+1}^*}\right) > 0$ or $P\left(\hat{\mu}^B_{a_K^*} \geq \hat{\mu}^*_{SE} > \hat{\mu}^B_{a_{K+1}^*}\right) > 0$. Moreover, $\forall K : 1 \leq K \leq N$, $\mathbb{E}\left[\min\left\{\hat{\mu}^B_{a_K^*}, \hat{\mu}^*_{SE}\right\}\right] \geq \mathbb{E}\left[\hat{\mu}^*_{CDE}\right]$.*

Notably, from the last inequality in Theorem 1, one can see that our estimator can effectively reduce the large underestimation bias in clipped double estimator. Moreover, since the existed inequality $\mathbb{E}\left[\hat{\mu}^*_{SE}\right] \geq \mathbb{E}\left[\min\left\{\hat{\mu}^B_{a_K^*}, \hat{\mu}^*_{SE}\right\}\right] \geq \mathbb{E}\left[\hat{\mu}^*_{CDE}\right]$, it essentially implies that the choice of K controls the trade-off between the overestimation bias in single estimator and the underestimation bias in clipped double estimator. For the proof please refer to Appendix A.

The upper bound of $\mathbb{E}\left[\hat{\mu}^*_{SE}\right]$ (van Hasselt 2013) is:

$$\mathbb{E}\left[\hat{\mu}^*_{SE}\right] = \mathbb{E}\left[\max_i \hat{\mu}_i\right] \leq \mu^* + \sqrt{\frac{N-1}{N}\sum_i^N \mathrm{Var}\left[\hat{\mu}_i\right]}. \tag{10}$$

Since $\mathbb{E}\left[\hat{\mu}^B_{a_K^*}\right]$ decreases monotonically as the number K increases (see Property 1 in Appendix A), $\mathbb{E}\left[\hat{\mu}^B_{a_1^*}\right]$ is maximum. Due to the candidate subset $\mathcal{M}_1$ only contains one candidate corresponding to the largest value in $\hat{\boldsymbol{\mu}}^B$, we can obtain $\mathbb{E}\left[\hat{\mu}^B_{a_1^*}\right] = \mathbb{E}\left[\max_i \hat{\mu}^B_i\right]$. Similar to the upper bound in $\mathbb{E}\left[\hat{\mu}^*_{SE}\right]$, we can see that $\mathbb{E}\left[\max_i \hat{\mu}^B_i\right] \leq \mu^* + \sqrt{\frac{N-1}{N}\sum_i^N \mathrm{Var}\left[\hat{\mu}^B_i\right]}$. Since $\hat{\mu}^B_i$ is just estimated via

Algorithm 2 Action Candidate Based TD3

```
Initialize critic networks $Q(\cdot;\boldsymbol{\theta}_1)$, $Q(\cdot;\boldsymbol{\theta}_2)$, and actor networks $\mu(\cdot;\boldsymbol{\phi}_1)$, $\mu(\cdot;\boldsymbol{\phi}_2)$ with random parameters $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$
Initialize target networks $\boldsymbol{\theta}_1^- \leftarrow \boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2^- \leftarrow \boldsymbol{\theta}_2$, $\boldsymbol{\phi}_1^- \leftarrow \boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2^- \leftarrow \boldsymbol{\phi}_2$
Initialize replay buffer $\mathcal{D}$
 1: for $t = 1 : T$ do
 2:     Select action with exploration noise $a \sim \mu(s;\boldsymbol{\phi}_1) + \epsilon$, $\epsilon \sim \mathcal{N}(0,\sigma)$ and observe reward $r$ and next state $s'$.
 3:     Store transition tuple $\langle s, a, r, s' \rangle$ in $\mathcal{D}$.
 4:     Sample a mini-batch of transitions $\{\langle s, a, r, s' \rangle\}$ from $\mathcal{D}$.
 5:     Determine $\boldsymbol{\mathcal{M}}_K = \{a_i\}_{i=1}^K, a_i \sim \mathcal{N}\left(\mu\left(s';\boldsymbol{\phi}_2^-\right), \bar{\sigma}\right)$ and define $a_K^* = \arg\max_{a\in\boldsymbol{\mathcal{M}}_K} Q\left(s', a; \boldsymbol{\theta}_1^-\right)$.
 6:     Update $\boldsymbol{\theta}_i \leftarrow \operatorname{argmin}_{\boldsymbol{\theta}_i} N^{-1} \sum \left[r + \gamma \min\left\{Q\left(s', a_K^*; \boldsymbol{\theta}_2^-\right), Q\left(s', \mu\left(s';\boldsymbol{\phi}_1^-\right); \boldsymbol{\theta}_1^-\right)\right\} - Q\left(s, a; \boldsymbol{\theta}_i\right)\right]^2$.
 7:     if $t$ mod $d$ then
 8:         Update $\boldsymbol{\phi}_i$ by the deterministic policy gradient: $\nabla_{\boldsymbol{\phi}_i} J(\boldsymbol{\phi}_i) = \frac{1}{N}\sum \nabla_a Q_{\theta_i}(s,a)\big|_{a=\mu(s;\boldsymbol{\phi}_i)} \nabla_{\boldsymbol{\phi}_i}\mu(s;\boldsymbol{\phi}_i)$.
 9:         Update target networks: $\boldsymbol{\theta}_i^- \leftarrow \tau\boldsymbol{\theta}_i + (1-\tau)\boldsymbol{\theta}_i^-$, $\boldsymbol{\phi}_i^- \leftarrow \tau\boldsymbol{\phi}_i + (1-\tau)\boldsymbol{\phi}_i^-$.
10:     end if
11: end for
```

S_i^B containing half of samples rather than S_i, $\mathrm{Var}\left[\hat{\mu}_i\right] \leq \mathrm{Var}\left[\hat{\mu}_i^B\right]$ and thus $\mu^* + \sqrt{\frac{N-1}{N}\sum_i^N \mathrm{Var}\left[\hat{\mu}_i\right]} \leq \mu^* + \sqrt{\frac{N-1}{N}\sum_i^N \mathrm{Var}\left[\hat{\mu}_i^B\right]}$. So, such larger upper bound may cause the maximum value $\mathbb{E}\left[\hat{\mu}_{a_1^*}^B\right]$ to exceed the $\mathbb{E}\left[\hat{\mu}_{SE}^*\right]$. Meanwhile, based on the monotonicity in Property 1, it further implies that when number K is too small, the upper of $\mathbb{E}\left[\hat{\mu}_{a_K^*}^B\right]$ tends to be larger than the one of $\mathbb{E}\left[\hat{\mu}_{SE}^*\right]$, which may cause larger overestimation bias. Therefore, the clipping operation guarantees that no matter how small the number of the selected candidates is, the overestimation bias of our estimator is no more than that of the single estimator.

Action Candidate Based Clipped Double Estimator for Double Q-learning and TD3

In this section, we apply our proposed action candidate based clipped double estimator into Double Q-learning and TD3. For the discrete action task, we first propose the action candidate based clipped Double Q-learning in the tabular setting, and then generalize it to the deep case with the deep neural network, that is action candidate based clipped Double DQN. For the continuous action task, we combine our estimator with TD3 and form action candidate based TD3.

Action Candidate Based Clipped Double Q-learning

Tabular Version In tabular setting, action candidate based Double Q-learning stores the Q-functions Q^A and Q^B, and learns them from two separate subsets of the online collected experience. Each Q-function is updated with a value from the other Q-function for the next state. Specifically, in order to update Q^A, we first determine the action candidates:

$$\boldsymbol{\mathcal{M}}_K = \left\{i | Q^B(s', a_i) \in \text{top } K \text{ values in } Q^B(s', \cdot)\right\}. \quad (11)$$

According to the action value function Q^A, the action a_K^* is the maximal valued action in the state s' among $\boldsymbol{\mathcal{M}}_K$. Then, we update Q^A via the target value as below:

$$y^{AC\_CDQ} = r + \gamma \min\left\{Q^B(s', a_K^*), \max_a Q^A(s', a)\right\}. \quad (12)$$

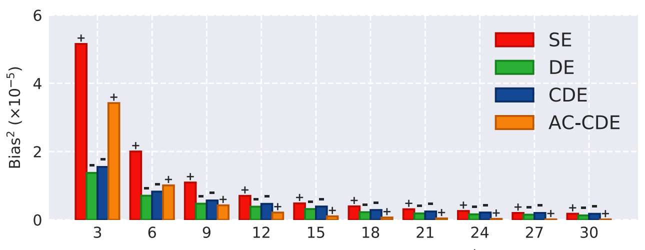
During the training process, the explored action is calculated with ϵ-greedy exploration strategy based on the action values Q^A and Q^B, More details are shown in Algorithm 1. Note that in the tabular version, the number of action candidates balances the overestimation in Q-learning and the underestimation in clipped Double Q-learning.

Deep Version For the task with the high-dimensional sensory input, we further propose the deep version of action candidate based clipped Double Q-learning, named action candidate based clipped Double DQN. In our framework, we maintain two deep Q-networks and an experience buffer. In each update, we independently sample a batch of experience samples to train each Q-network with the target value in Eq. 12. Similar to the tabular version, the number of action candidates can also balance the overestimation in DQN and the underestimation in clipped Double DQN. In addition, we verify that the action candidate based clipped Double Q-learning can converge to the optimal policy in the finite MDP setting. The proof can be seen in Appendix B.

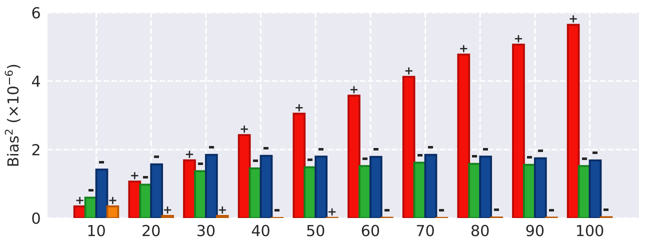
Action Candidate Based TD3

As shown in Algorithm 2, the algorithm framework for the continuous action task follows the design in TD3. To approximate the optimal action values, we construct two Q-networks $Q(s, a; \boldsymbol{\theta}_1)$ and $Q(s, a; \boldsymbol{\theta}_2)$ and two target Q-networks $Q\left(s, a; \boldsymbol{\theta}_1^-\right)$ and $Q\left(s, a; \boldsymbol{\theta}_2^-\right)$. In addition, two deterministic policy networks $\mu(s;\boldsymbol{\phi}_1)$ and $\mu(s;\boldsymbol{\phi}_2)$, and two target networks $\mu\left(s;\boldsymbol{\phi}_1^-\right)$ and $\mu\left(s;\boldsymbol{\phi}_2^-\right)$ are exploited to represent the optimal decisions corresponding to $Q(s, a; \boldsymbol{\theta}_1)$, $Q(s, a; \boldsymbol{\theta}_2)$, $Q\left(s, a; \boldsymbol{\theta}_1^-\right)$ and $Q\left(s, a; \boldsymbol{\theta}_2^-\right)$.

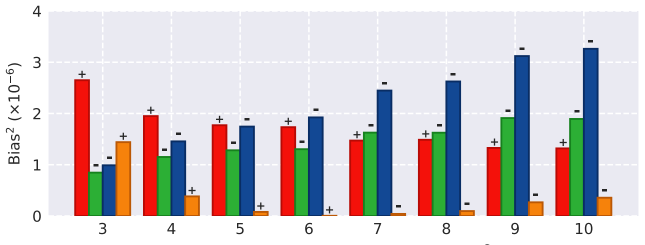
Due to the continuity of the actions, it is impossible to precisely determine the top K action candidates $\boldsymbol{\mathcal{M}}_K$ like in the discrete action case. We first exploit our deterministic policy network $\mu\left(s';\boldsymbol{\phi}_2^-\right)$ to approximate the global optimal action $a^* = \arg\max_a Q\left(s', a; \boldsymbol{\theta}_2^-\right)$. Based on the estimated global optimal action a^*, we randomly select K



(a) Number of impressions ($\times 10^4$)



(b) Number of ads



(c) Max probability in range ($\times 10^{-2}$)

Figure 1: Comparison on the multi-armed bandits for internet ads in three cases: (a) Varying the number of impressions; (b) Varying the number of ads; (c) Varying the max probability. The symbol on the bar represents the sign of the bias. The results are averaged over $2,000$ experiments. We use 15% of actions as the action candidates.

actions $\boldsymbol{\mathcal{M}}_K$ in the δ-neighborhood of a^* as the action candidates. Specifically, we draw K samples from a Gaussian distribution $\mathcal{N}\left(\mu\left(s';\boldsymbol{\phi}_2^-\right),\bar{\sigma}\right)$:

$$\boldsymbol{\mathcal{M}}_K = \left\{a_i | a_i \sim \mathcal{N}\left(\mu\left(s';\boldsymbol{\phi}_2^-\right),\bar{\sigma}\right), i = 1, 2, \dots, K\right\}, \quad (13)$$

where the hyper-parameter $\bar{\sigma}$ is the standard deviation. Both Q-networks are updated via the following target value:

$$y^{AC\_TD3} = r + \gamma \min\left\{Q\left(s', a_K^*; \boldsymbol{\theta}_2^-\right), Q\left(s', \mu\left(s'; \boldsymbol{\phi}_1^-\right); \boldsymbol{\theta}_1^-\right)\right\}, \quad (14)$$

where $a_K^* = \arg\max_{a \in \boldsymbol{\mathcal{M}}_K} Q\left(s', a; \boldsymbol{\theta}_1^-\right)$. The parameters of two policy networks are updated along the direction that can improve their corresponding Q-networks. For more details please refer to Algorithm 2.

Experiments

In this section, we empirically evaluate our method on the discrete and continuous action tasks.

For the discrete action tasks, we conduct the following three experiments:

- For action candidate based clipped double estimator (AC-CDE), we compare them with single estimator (Hasselt 2010), double estimator (Hasselt 2010) and clipped double estimator (Fujimoto, Van Hoof, and Meger 2018) on the multi-armed bandits problem.
- For action candidate based clipped Double Q-learning (AC-CDQ), we compare them with Q-learning (Watkins and Dayan 1992), Double Q-learning (Hasselt 2010) and clipped Double Q-learning (Fujimoto, Van Hoof, and Meger 2018) on grid world game.
- For action candidate based clipped Double DQN (AC-CDDQN), we compare them with DQN (Mnih et al. 2015), Double DQN (Van Hasselt, Guez, and Silver 2016), Averaged-DQN (Anschel, Baram, and Shimkin 2017) and clipped Double DQN (Fujimoto, Van Hoof, and Meger 2018) on several benchmark games in MinAtar (Young and Tian 2019).

For the continuous action tasks, we compare our action candidate based TD3 (AC-TD3) with TD3 (Fujimoto, Van Hoof, and Meger 2018), SAC (Haarnoja et al. 2018) and DDPG (Lillicrap et al. 2015) on six MuJoCo (Todorov, Erez, and Tassa 2012) based benchmark tasks implemented in OpenAI Gym (Dhariwal et al. 2017).

Discrete Action Tasks

Multi-Armed Bandits For Internet Ads In this experiment, we employ the framework of the multi-armed bandits to choose the best ad to show on the website among a set of M possible ads, each one with an unknown fixed expected return per visitor. For simplicity, we assume each ad has the same return per click, such that the best ad is the one with the maximum click rate. We model the click event per visitor in each ad i as the Bernoulli event with mean m_i and variance $(1 - m_i)m_i$. In addition, all ads are assumed to have the same visitors, which means that given N visitors, N/M Bernoulli experiments will be executed to estimate the click rate of each ad. The default configuration in our experiment is $N = 30,000$, $M = 30$ and the mean click rates uniformly sampled from the interval $[0.02, 0.05]$. Based on this configuration, there are three settings: (1) We vary the number of visitors $N = \{30,000, 60,000, \dots, 270,000, 300,000\}$. (2) We vary the number of ads $M = \{10, 20, \dots, 90, 100\}$. (3) We vary the upper limit of the sampling interval of mean click rate (the original is 0.05) with values $\{0.03, 0.04, \dots, 0.09, 0.1\}$.

To compare the absolute bias, we evaluate the single estimator, double estimator, clipped double estimator and AC-CDE with the square of bias ($bias^2$) in each setting. As shown in Fig. 1, compared to other estimators, AC-CDE owns the lowest $bias^2$ in almost all experimental settings. It mainly benefits from the effective balance of our proposed estimator between the overestimation bias of single estimator and underestimation bias of clipped double estimator. Moreover, AC-CDE has the lower $bias^2$ than single estimator in all cases while in some cases it has the larger $bias^2$ than clipped double estimator such as the leftmost columns in Fig. 1 (a) and (c). It's mainly due to that although AC-CDE can reduce the underestimation bias of clipped double

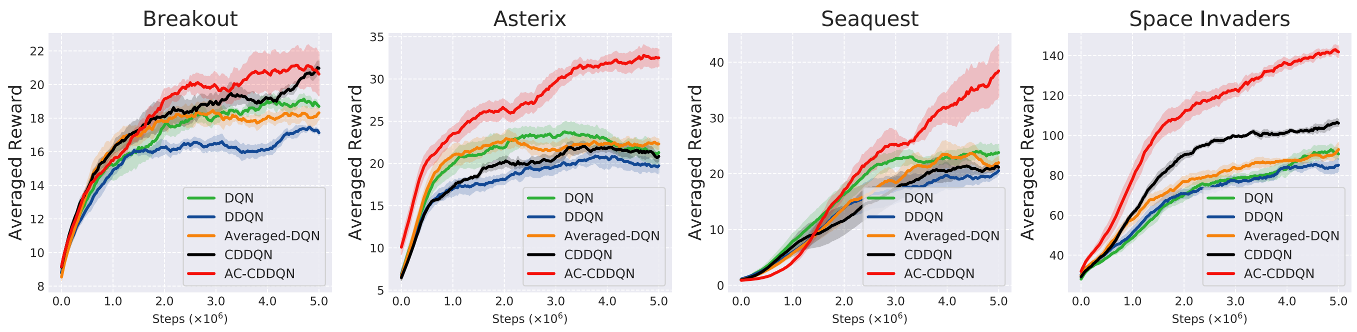


Figure 2: Learning curves on the four MinAtar benchmark environments. The results are averaged over five independent learning trials and the shaded area represents half a standard deviation.

estimator, too small number of action candidates may also in turn cause overestimation bias (no more than single estimator). Thus, the absolute value of such overestimation bias may be larger than the one of the underestimation bias in clipped double estimator. Despite this, AC-CDE can guarantee that the positive bias is no more than single estimator and the negative bias is also no more than clipped double estimator.

Grid World As a MDP task, in a $N \times N$ grid world, there are total N^2 states. The starting state s_0 is in the lower-left cell and the goal state is in the upper-right cell. Each state has four actions: east, west, south and north. At any state, moving to an adjacent cell is deterministic, but a collision with the edge of the world will result in no movement. Taking an action at any state will receive a random reward which is set as below: if the next state is not the goal state, the random reward is -6 or $+4$ and if the agent arrives at the goal state, the random reward is -30 or $+40$. With the discount factor γ, the optimal value of the maximum value action in the starting state s_0 is $5\gamma^{2(N-1)} - \sum_{i=0}^{2N-3}\gamma^i$. We set N to 5 and 6 to construct our grid world environments and compare the Q-learning, Double Q-learning, clipped Double Q-learning and AC-CDQ ($K = 2, 3$) on the mean reward per step and estimation error (see Fig. 3).

From the top plots, one can see that AC-CDQ ($K = 2, 3$) can obtain the higher mean reward than other methods in both given environments. We further plot the estimation error about the optimal state value $V^*(s_0)$ in bottom plots. Compared to Q-learning, Double Q-learning and clipped Double Q-learning, AC-CDQ ($K = 2, 3$) show the much lower estimation bias (more closer to the dash line), which means that it can better assess the action value and thus help generate more valid action decision. Moreover, our AC-CDQ can significantly reduce the underestimation bias in clipped Double Q-learning. Notably, as demonstrated in Theorem 1, the underestimation bias in the case of $K = 2$ is smaller than that in the case of $K = 3$. And AC-CDQ can effectively balance the overestimation bias in Q-learning and the underestimation bias in clipped Double Q-learning.

MinAtar MinAtar is a game platform for testing the reinforcement learning algorithms, which uses a simplified state representation to model the game dynamics of

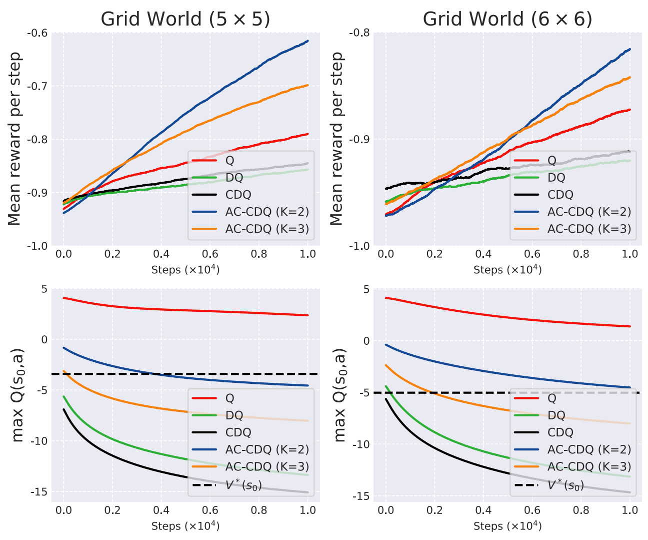


Figure 3: The top plots show the mean reward per step and the bottom plots show the estimated maximum action value from the state s_0 (the black dash line demotes the optimal state value $V^*(s_0)$). The results are averaged over 10000 experiments and each experiment contains 10000 steps. We set the number of the action candidates to 2 and 3, respectively.

Atari from ALE (Bellemare et al. 2013). In this experiment, we compare the performance of DQN, Double DQN, Averaged-DQN, clipped Double DQN and AC-CDDQN on four released MinAtar games including Breakout, Asterix, Seaquest and Space Invaders. We exploit the convolutional neural network as the function approximator and use the game image as the input to train the agent in an end-to-end manner. Following the settings in (Young and Tian 2019), the hyper-parameters and settings of neural networks are set as follows: the batch size is 32; the replay memory size is 100,000; the update frequency is 1; the discounted factor is 0.99; the learning rate is 0.00025; the initial exploration is 1; the final exploration is 0.1; the replay start size is 5,000. The optimizer is RMSProp with the gradient momentum 0.95 and the minimum squared gradient 0.01. The experimental results are obtained after 5M frames.

Fig. 2 represents the training curve about averaged reward

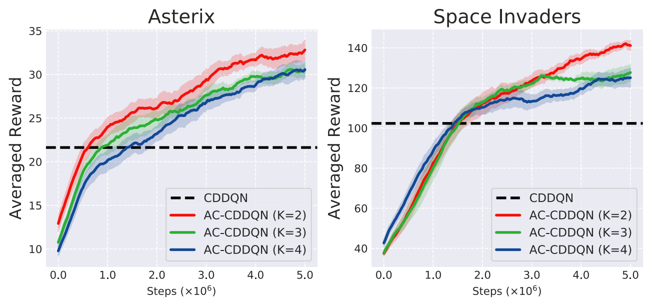


Figure 4: Learning curves on the several MinAtar benchmark environments for AC-CDDQN with different numbers of the action candidates.

of each algorithm. It shows that compared to DQN, Double DQN Averaged-DQN and clipped Double DQN, AC-CDDQN can obtain better or comparable performance while they have the similar convergence speeds in all four games. Especially, for Asterix, Seaquest and Space Invaders, AC-CDDQN can achieve noticeably higher averaged rewards compared to the clipped Double DQN and obtain the gains of 36.3%, 74.4% and 19.8%, respectively. Such significant gain mainly owes to that AC-CDDQN can effectively balance the overestimation bias in DQN and the underestimation bias in clipped Double DQN. Moreover, in Fig. 4 we also test the averaged rewards of different numbers of action candidates $K = \{2, 3, 4\}$ for AC-CDDQN. The plots show that AC-CDDQN is consistent to obtain the robust and superior performance with different action candidate sizes.

Continuous Action Task

MuJoCo Tasks We verify our variant for continuous action, AC-TD3, on six MuJoCo continuous control tasks from OpenAI Gym including Ant-v2, Walker2D-v2, Swimmer-v2, Pusher-v2, Hopper-v2 and Reacher-v2. We compare our method against the DDPG and two state of the art methods: TD3 and SAC. In our method, we exploit the TD3 as our baseline and just modify it with our action candidate mechanism. The number of the action candidate is set to 32. We run all tasks with 1 million timesteps and the trained policies are evaluated every $5,000$ timesteps.

We list the training curves of Ant-v2 and Swimmer-v2 in the top row of Fig. 5 and more curves are listed in Appendix C. The comprehensive comparison results are listed in Table 1. From Table 1, one can see that DDPG performs poorly in most environments and TD3 and SAC can't handle some tasks such as Swimmer-v2 well. In contrast, AC-TD3 consistently obtains the robust and competitive performance in all environments. Particularly, AC-TD3 owns comparable learning speeds across all tasks and can achieve higher averaged reward than TD3 (our baseline) in most environments except for Hopper-v2. Such significant performance gain demonstrates that our proposed approximate action candidate method in the continuous action case is effective empirically. Moreover, we also explain the performance advantage of our AC-TD3 over TD3 from the perspective of the bias (see the bottom row of the Fig. 5). The bottom plots show that in Ant-v2 and Swimmer-v2, AC-TD3 tends to have a lower estimation bias than TD3 about the expected return with regard to the initial state s_0, which potentially helps the agent assess the action at some state better and then generate the more reasonable policy.

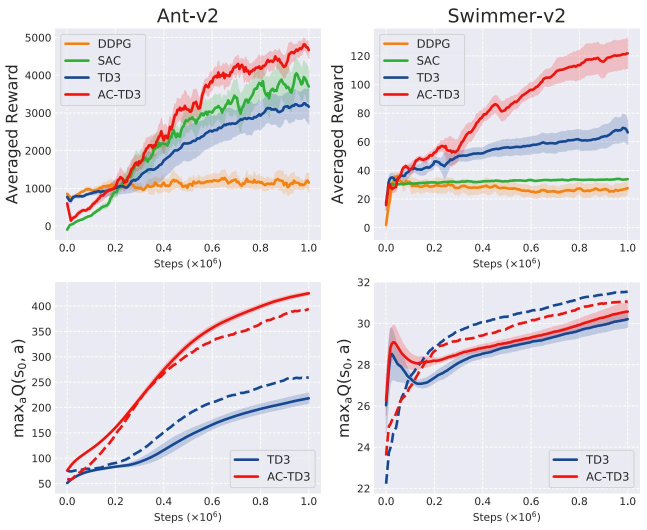


Figure 5: Top row: Learning curves for the OpenAI Gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Bottom plots: the estimation of the expected return with respect to the initial state s_0 of the game. The dash lines represent the real discounted return.

Table 1: Averaged rewards over last 30% episodes during training process.

	AC-TD3	TD3	SAC	DDPG
Pusher-v2	**-22.70 $\pm$ 0.39**	-31.81	-76.78	-38.40
Reacher-v2	**-3.57 $\pm$ 0.06**	**-3.62**	-12.92	-8.98
Walker2d-v2	**3800.30 $\pm$ 130.95**	3530.41	1863.81	1849.98
Hopper-v2	2827.23 $\pm$ 83.23	2974.86	**3111.05**	2611.44
Swimmer-v2	**116.19 $\pm$ 3.63**	63.18	33.55	30.17
Ant-v2	**4391 $\pm$ 205.61**	3044.61	3646.46	1198.63

Conclusion

In this paper, we proposed an action candidate based clipped double estimator to approximate the maximum expected value. Furthermore, we applied this estimator to form the action candidate based clipped Double Q-learning. Theoretically, the underestimation bias in clipped Double Q-learning decays monotonically as the number of action candidates decreases. The number of the action candidates can also control the trade-off between the overestimation and underestimation. Finally, we also extend our clipped Double Q-learning to the deep version and the continuous action tasks. Experimental results demonstrate that our proposed methods yield competitive performance.

References

Anschel, O.; Baram, N.; and Shimkin, N. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *ICML*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *JAIR* 47: 253–279.

Bellman, R. 1952. On the theory of dynamic programming. *PNAS* 38(8): 716.

D'Eramo, C.; Restelli, M.; and Nuara, A. 2016. Estimating maximum expected value through gaussian approximation. In *ICML*.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. Openai baselines.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. *JMLR* 6(Apr): 503–556.

Fujimoto, S.; Van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* .

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* .

Hasselt, H. V. 2010. Double Q-learning. In *NIPS*.

Lan, Q.; Pan, Y.; Fyshe, A.; and White, M. 2020. Q-learning. *arXiv preprint arXiv:2002.06487* .

Lee, D.; Defourny, B.; and Powell, W. B. 2013. Bias-corrected Q-learning to control max-operator bias in Q-learning. In *ADPRL*.

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

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

Strehl, A. L.; Li, L.; and Littman, M. L. 2009. Reinforcement learning in finite MDPs: PAC analysis. *JMLR* 10(Nov): 2413–2444.

Strehl, A. L.; Li, L.; Wiewiora, E.; Langford, J.; and Littman, M. L. 2006. PAC model-free reinforcement learning. In *ICML*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*.

Szita, I.; and Lőrincz, A. 2008. The many faces of optimism: a unifying approach. In *ICML*.

Thrun, S.; and Schwartz, A. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *IROS*.

van Hasselt, H. 2013. Estimating the maximum expected value: an analysis of (nested) cross validation and the maximum sample average. *arXiv preprint arXiv:1302.7175* .

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

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

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

Zhang, Z.; Pan, Z.; and Kochenderfer, M. J. 2017. Weighted Double Q-learning. In *IJCAI*.