Successively Pruned Q-Learning: Using Self Q-function to Reduce the Overestimation

Zhaolin Xue
Fudan University
Shanghai, China
21210860021@m.fudan.edu.cn

Lihua Zhang
Fudan University
Shanghai, China
lihuazhang@fudan.edu.cn

Zhiyan Dong *
Fudan University
Shanghai, China
dongzhiyan@fudan.edu.cn

ABSTRACT

It's well-known that the Q-learning algorithm suffers the overestimation owing to using the maximum state-action value as an approximation of the maximum expected state-action value. Double Q-learning and other algorithms have been proposed as efficient solutions to alleviate the overestimation. However, these proposed methods intend to utilize multiple Q-functions to reduce the overestimation and ignore the information of single Q-function. In this paper, 1) we reinterpret the update process of Q-learning, build a more precise model compatible with previous model. 2) We propose a novel and simple method to control the maximum bias by employing the information of single Q-function. 3) Our method not only balances between the overestimation and the underestimation, but also attains the minimum bias under proper hyper-parameters. 4) Moreover, it can be naturally generalized to the discrete control domain and continuous control tasks. We reveal that our algorithms outperform Double DQN and other algorithms on some representative games and some classical off-policy actor-critic algorithms can also gain benefits from our method.

KEYWORDS

Q-learning; Double Q-learning; Overestimation; DQN; DDPG; TD3; SAC

ACM Reference Format:
Zhaolin Xue, Lihua Zhang, and Zhiyan Dong *. 2024. Successively Pruned Q-Learning: Using Self Q-function to Reduce the Overestimation. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Markovian decision problem is a mathematical model of temporal credit assignment problems and usually studied in stochastic control theory [2]. Reinforcement Learning (RL) aims at learning how to map situations to actions as to maximize a cumulative reward [33]. One of the most popular model-free algorithms for optimally solving the above problem is the Q-learning algorithm introduced by [38]. Q-learning has been proven that can converge to optimal value in tabular setting no matter in discounted case or non-discounted case [16, 38]. Moreover, Q-learning can be extended to the deep neural network model which can directly map high-dimensional inputs to estimated state-action value [23]. The rapid development of Q-learning has accelerated the deployment on practical RL problems such as navigation [25] and control [40].

However in practice, the performance of Q-learning can be poor in stochastic MDPs due to the positive bias against the true value. To mitigate the problem, Thrun et al. [34] analysed that $E\{\max_i Q(s, a_i)\} \geq \max_i E\{Q(s, a_i)\}$ ($i \in [1, M]$) and based on that, Hasselt et al. [15] proposed an alternative approach called the double estimator dividing the process of choosing approximated maximum expected value estimation into two stages: action section stage and state-action value estimation stage. Besides, other algorithms are presented to utilize multiple Q-functions to accurately estimate Q value such as [13, 19, 20, 24, 27, 37, 41]. Although multiple Q-functions indeed improve the accuracy, splitting up all samples into multiple sets causes the waste of resources and slows the convergence speed. Based on that, we wonder whether the previous values can be utilized to solve the overestimation and improve the convergence speed and value. Hence, in this paper, we will discuss how to use previous values to mitigate the overestimation in the following sections. In Section 2, we introduce the basic background of our method. In Section 3, we analyse the maximization bias of Q-learning from two perspectives containing maximum expected value estimation and averaged maximum expected value estimation. In Section 4, we propose a novel but simple algorithm called *successively pruned q-learning* which can solve the overestimation bias and we demonstrate that our method can achieve minimum bias under appropriate parameters and convergence to the optimal value theoretically. In Section 5, *successively pruned q-learning* algorithm is extended to deep neural network domain. In Section 6, *successively pruned q-learning* and its generalization of off-policy algorithms are examined on several experiments and the results are shown through figures and tables. In Section 7, we conclude this paper and propose future directions. In this paper, the main contributions we make are as follows:

- We construct a serial model including the error against the iteration time n and the common error shared by all state-action Q values.
- Based on the proposed model, we present a simple but effective method named *successively pruned q-learning* to balance between the overestimation and the underestimation and we demonstrate that under proper hyper-parameters, this method can attain the minimum bias against the true value.
- This method is triumphantly stretched to deep neural network domain [23]. Successively pruned DQN presented still appears the superiority over other algorithms and some classical off-policy actor-critic algorithms with successively pruned tool outperform original algorithms.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Related Work. A growing number of algorithms are inspired to promote the Q-learning algorithm in various aspects such as convergence speed and value estimation. Fitted Q-iteration [12], Delayed Q-learning [31] and Speedy Q-learning [4] have been proposed to accelerate convergence speed of the Q-learning algorithm. As for the value estimation, many algorithms tend to reduce the variance and the expected value of maximization bias. Averaged-DQN [1], TD3 [13] and TQC [19] have mitigated the instability and variability of Deep Reinforcement Learning (DRL) algorithms. Some algorithms aim at alleviating the expected value of maximization bias under specific assumptions like the Gaussian approximation [9, 11]. Others concentrate on balancing between the overestimation and the underestimation such as [21, 41, 42]. Zhang et al. [41] propose to weight Q-learning and Double Q-learning to realize the balance and Zhu et al. [42] present a self-correcting q-learning method to weight Q_{n-1} and Q_n. In contrast with our method, self-correcting q-learning omits the construction of Q_{n-1}, generating serious underestimation and fails to be extended to the continuous control domain. Several algorithms continue to study the properties of double q-learning like [7, 17, 27, 39]. Chen et al. [7] also use two critic Q-functions but decorrelate these two value functions to further reduce the overestimation. Weng et al. [39] formulate Double Q-learning and Q-learning as instances of Linear Stochastic Approximation (LSA) by a linearization technique in [10]. Jiang et al. [17] devise an action candidate method based on the clipped double estimator [13] to approximate the maximum expected value. Ren et al. [27] reveal that underestimation bias may leads to multiple non-optimal fixed points under an approximate Bellman operator.

Besides, ensemble methods [20, 24, 37] tackle this problem by diminishing the MSE of the next state-action value. Maxmin Q-learning [20] utilizes multiple Q-functions to approximate the true value. Peer et al. [24] demonstrate that the mean-squared-error (MSE) of estimation can be reduced through the ensemble estimator. Adaptive Ensemble Q-learning [37] leverages Model Identification Adaptive Control (MIAC) [3] to control the ensemble size. Nevertheless, as we known, these aforementioned methods neglect to analyse the relation between Q_n and $\{Q_{n-1}, Q_{n-2}, Q_{n-3} \ldots\}$. Therefore, in the following sections, we supplement the theoretical research and experimental study.

2 MARKOV DECISION PROCESS

Markov decision process (MDP) is the problem that an agent acts in a stochastic environment by sequentially choosing actions over a sequence of time steps to maximize a cumulative reward. Usually, the MDP framework consists of a five-tuple model (S, A, P, R, γ) and a policy π, where S is a finite set of states, A is a finite set of actions, P is a state transition distribution where $P_{ss'}(a_i)$ represents the probability distribution of transitioning from state s to state s' (next state) after executing action a_i (i denotes the index of a, $i \in [1, M]$, $a_i \in A$ and M represents the number of A), R is a reward function and the agent gains a reward when the agent transitions from state s to state s' after executing action a_i, and $\gamma \in [0, 1)$ is a discount factor balancing between instant and future rewards. The policy π is a mapping from state s to action a_i.

We define a value function which is the expected sum of discounted rewards beginning in state s and executing policy π:

$$V_\pi(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | s_0 = s\right\} \tag{1}$$

The relation between the optimal state value and the optimal state-action value can be shown:

$$\begin{aligned} V^*(s) &= \max_i Q^*(s, a_i) \\ Q^*(s, a_i) &= R(s, a_i) + \gamma \sum_{s' \in S} P_{ss'}(a_i) \max_{i'} Q^*(s', a_{i'}) \end{aligned} \tag{2}$$

Among the equation, $i' \in [1, M]$, $R(s, a_i) = \sum_{s' \in S} P_{ss'}(a_i) R(s, s', a_i)$ and the optimal state-action function accords with Bellman's equation. To find out the optimal value, the Q-learning algorithm is proposed to estimate the optimal state-action value [38].

3 ANALYSIS OF MAXIMIZATION BIAS

In this section, we analyse the phenomenon of maximization bias in Q-learning from two perspectives. The first perspective is to estimate the maximum expected value estimation and almost all current algorithms aim to find an approximation for $E\{\max_i Q(s, a_i)\}$. Besides, we propose a novel perspective concerning averaged maximum expected value to estimate the maximization bias. In this perspective, we decompose Q_{n+1} to all successive values to reveal the cumulative error during the update process.

3.1 Maximum expected value estimation

Suppose that an agent can execute M actions and then there are a set of M random variables $\{Q(s, a_1), \ldots, Q(s, a_M)\}$. The goal is to find the $\max_i E\{Q(s, a_i)\}$ without any assumptions on underlying distributions. Q-learning uses $E\{\max_i Q(s, a_i)\}$ to approximately replace the maximum expected value estimation. However, this method is positively biased since $E\{\max_i \{Q(s, a_i)\} \geq \max_i E\{Q(s, a_i)\}$ from Jensen's inequality. Focusing on mitigating the bias, many algorithms propose various alternative estimations [7, 21, 39, 41, 42].

3.2 Averaged maximum expected value estimation

In tabular setting, the update process of Q-learning is shown as follows:

$$\begin{aligned} Q_{n+1}(s, a_i) = &(1 - \alpha_n(s, a_i)) Q_n(s, a_i) \\ &+ \alpha_n(s, a_i)[r_n + \gamma \max_{i'} Q_n(s', a_{i'})] \end{aligned} \tag{3}$$

where s' denotes the next state, $i \in [1, M]$, $i' \in [1, M]$ and M represents the number of actions. In most cases, $\alpha_n(s, a_i)$ equals to $\frac{1}{n+1}$. Simplifying Equ (3) by the definition of the $\alpha_n(s, a_i)$ and we get the following equation:

$$\begin{aligned} F_n &= r_n + \gamma \max_{i'} Q_n(s', a_{i'}) \\ Q_{n+1}(s, a_i) &= (1 - \frac{1}{n+1}) Q_n(s, a_i) + \frac{1}{n+1} F_n \\ &= \frac{F_0 + F_1 + \cdots + F_n}{n+1} \end{aligned} \tag{4}$$

which illustrates that Q_{n+1} is determined by averaged F_l, in which $l \in [0, n]$. The randomness of F_l leads to the stochasticity of $Q_{n+1}(s, a_i)$. To conveniently understand accumulative error, we rectify the following assumption:

$$Q_n(s, a_i) = Q^*(s, a_i) + o \tag{5}$$

where o is a uniform random variable $U(-\tau, \tau)$ for some $\tau > 0$. This assumption was proposed by Thrun et al. [34] and employed by [7, 15, 20, 41, 42]. But it is improper if we directly assume $\{Q_n, Q_{n-1}, Q_{n-2}, \ldots\}$ have the same error o. Because we perceive that $Q_n(s, a_i)$ approaches to the optimal value constantly with the increase of iteration times n. Hence, the iteration times n is a significant factor in influencing the Q value. We extend the error model and augment a uniform random variable $\epsilon_j \sim U(-\eta_j, \eta_j)$ for $j \in [1, n]$ and $\eta_j \geq 0$ to reflect the effect of iteration time. The final model is as follows:

$$\begin{aligned} Q_j(s, a_i) &= Q^*(s, a_i) + \delta_{ji} \\ \delta_{ji} &= \delta_j = o + \epsilon_j \end{aligned} \tag{6}$$

where a_i denotes the selected action, $i \in [1, M]$, $j \in [1, n]$ and o is a common random error shared by all Q and mentioned in Equ (5). We analyse the mathematical model including *probability density function* (PDF) $f_{\delta_j}(x)$ and *cumulative distribution function* (CDF) $F_{\delta_j}(x)$ in Appendix A.1. When $\eta_j = 0$, Equ (6) become vestigial to Equ (5).

We can get the following equation by utilizing Equ (2,4,6):

$$\begin{aligned} E\{Q_n(s, a_i)\} - Q^*(s, a_i) &= \gamma \frac{\sum_{j=0}^{n-1} E\{\max_{i'} Q_j(s', a_{i'}) - \max_{i''} Q^*(s', a_{i''})\}}{n} \\ &\leq \gamma \frac{\sum_{j=0}^{n-1} E\{\max_{i'''}(Q_j(s', a_{i'''}) - Q^*(s', a_{i'''}))\}}{n} \\ &\leq \gamma \frac{\sum_{j=0}^{n-1} E\{\max_{i'''}(\delta_{ji'''})\}}{n} \end{aligned} \tag{7}$$

where s' denotes the next state from state s, $i', i'', i''' \in [1, M]$ and $a_{i'}, a_{i''}, a_{i'''}$ denote the actions derived from the next state. Similarly, we can get the following equations combining Equ (2,4,6):

$$\begin{aligned} i''_* &= \operatorname*{argmax}_{i''} Q^*(s', a_{i''}) \\ E\{Q_n(s, a_i)\} - Q^*(s, a_i) &= \gamma \frac{\sum_{j=0}^{n-1} E\{\max_{i'} Q_j(s', a_{i'}) - Q^*(s', a_{i''_*})\}}{n} \\ &\geq \gamma \frac{\sum_{j=0}^{n-1} E\{Q_j(s', a_{i''_*}) - Q^*(s', a_{i''_*})\}}{n} \\ &\geq 0 \end{aligned} \tag{8}$$

Clearly, according to Equ (7,8), we have that:

$$0 \leq E\{Q_n(s, a_i)\} - Q^*(s, a_i) \leq \gamma \sum_{j=0}^{n-1} E\{\max_{i'''}(\delta_{ji'''})/n \tag{9}$$

Therefore, the bias of Q-learning is bounded by the accumulated error $\sum_{j=0}^{n-1} E\{\max_{i'''}(\delta_{ji'''})/n$. We specifically let $\alpha_n(s, a_i) = \frac{1}{n+1}$ above, and the condition can be extended to $\sum_{j=0}^{n-1} \prod_{k=j+1}^{n-1} (1-\alpha_k)\alpha_j E\{\max_{i'''}(\delta_{ji'''})\}$ when α_n is unknown. Obviously, $\sum_{j=0}^{n-1} \prod_{k=j+1}^{n-1} (1-\alpha_k)\alpha_j = 1$ and we can modify α to satisfy demand for different weights.

4 SUCCESSIVELY PRUNED Q-LEARNING

In this section, a novel algorithm called *successively pruned q-learning* is proposed to reduce the bias. According to the aforementioned analysis, we know that the bias of $E\{Q_n(s, a_i)\}$ is bounded by the accumulated error $\sum_{j=0}^{n-1} E\{\max_{i'''}(\delta_{ji'''})/n$, meaning the bias is influenced by the historical information, which inspires us to explore whether historical information can be utilized to reduce the overestimation bias. We notice that if the Q-learning algorithm overestimates when $n = r$, and then the chain effect of sequential updates would lead to the amplification of overestimation. To control the sequential error, we propose to minimize through the historical values, which denotes that firstly selecting the action and secondly choosing the value estimation:

$$\begin{aligned} i'_* &= \operatorname*{argmax}_{i'} Q_n(s', a_{i'}) \\ Y_n^K &= \min(\underbrace{Q_n(s', a_{i'_*}), Q_{n-1}(s', a_{i'_*}), Q_{n-2}(s', a_{i'_*}), \ldots)}_{K} \end{aligned} \tag{10}$$

where s' denotes the next state from the state s and K represents the parameter controlling the number of joining in the minimum. The complete algorithm is shown in Algorithm 1.

4.1 Bias analysis of *successively pruned q-learning*

Next, we analyse the bias between mean value estimation of *successively pruned q-learning* and the true value. The bias satisfies:

$$\begin{aligned} E\{B_n^K\} &= E\{Y_n^K\} - E\{\max_{i''} Q^*(s', a_{i''})\} \\ S_1 &= \int_{-\infty}^{+\infty} x f_{M:M}(x) \prod_{l=1}^{K-1} (1 - F_{\delta_{n-l}}(x)) dx \\ S_2 &= \int_{-\infty}^{+\infty} x(1 - F_{M:M}(x)) \sum_{p=1}^{K-1} f_{\delta_{n-p}}(x) \prod_{q=1, q\neq p}^{K-1} (1 - F_{\delta_{n-q}}(x)) dx \\ E\{B_n^K\} &= E\{\min(\underbrace{\max(\delta_{n1}, \ldots, \delta_{nM}), \delta_{n-1}, \ldots)}_{K}\} = S_1 + S_2 \end{aligned} \tag{11}$$

where $f_{M:M}(x) = M f_{\delta_n}(x)(F_{\delta_n}(x))^{M-1}$ and $F_{M:M}(x) = (F_{\delta_n}(x))^M$, respectively representing the PDF and CDF of $\max(\delta_{n1}, \ldots, \delta_{nM})$. The specific definition of $f_\delta(x)$ and $F_\delta(x)$ can refer to Appendix A.1. Obviously, Equ (11) is hard to calculate and we can simplify the analysis process. First of all, we analyse the situation of $K = 1$. We know that the first item $E\{\min(\max(\delta_{n1}, \ldots, \delta_{nM}))\} = E\{\max(\delta_{n1}, \ldots, \delta_{nM})\}$. And according to lemma 1 in Appendix

A.2, we get that $0 < E\{\max(\delta_{n1}, \ldots, \delta_{nM})\} < \eta_n + \tau$. Therefore, $E\{B_n^1\} > 0$. For $K > 1$, let $Z_n = \max(\delta_{n1}, \ldots, \delta_{nM})$ and $Z_{n-w} = \delta_{n-w}, w \in [1, K-1]$. Define j^* representing the index of minimum item and $j^* \in [n-K+1, n]$. The following formulas are satisfied:

$$\begin{aligned} E\{B_n^K\} &= P(j^* = n-K+1)E\{B_n^K | P(j^* = n-K+1)\} \\ &\quad + P(j^* \neq n-K+1)E\{B_n^K | P(j^* \neq n-K+1)\} \\ &\leq P(j^* = n-K+1)E\{B_n^{K-1} | P(j^* = n-K+1)\} \\ &\quad + P(j^* \neq n-K+1)E\{B_n^{K-1} | P(j^* \neq n-K+1)\} \\ &\leq E\{B_n^{K-1}\} \end{aligned} \tag{12}$$

where the inequality is strict if and only if $P(j^* = n-K+1) > 0$. Next, we just need to give a special example to demonstrate the existence of $j^* = n-K+1$. Because $\{Z_n, \ldots, Z_{n-K+1}\}$ are independent with each other, we consider a situation where $Z_n = \ldots = Z_{n-K+2} = 0$ and $Z_{n-K+1} \in [-\eta_{n-K+1} - \tau, 0)$. Obviously, in that situation, j^* equals to $n-K+1$. Therefore, the inequality is strict:

$$E\{B_n^K\} < E\{B_n^{K-1}\} \tag{13}$$

In summary, the bias is the overestimation when $K = 1$ and with the increase of K, the bias strictly decreases. Hence, it is existent that $E\{B_n^{K^*}\} \geq 0$ and $E\{B_n^{K^*+1}\} \leq 0$ for $K = K^*$. This implies that our algorithms attains the minimum bias within $E\{B_n^K\}$. More details can be found in Appendix A.

4.2 Convergence of *successively pruned q-learning*

This part is to demonstrate the convergence of *successively pruned q-learning*. Firstly, we present lemma 1 and then lemma 2 can be proved based on the lemma 1 and some lemmas in [16]. Finally, lemma 2 is utilized to directly prove Theorem 1 without going into technical details.

Lemma 1. *Consider a random iterative process* $X_{n+1}(x) = (1-\alpha_n(x))X_n(x) + \gamma\alpha_n(x)(\|X_n\| + \max_t \|X_n - X_{n-t}\|)$ $(t \in [0, K-1], K > 0)$ *converges to zero w.p.1 if provided the following conditions:* $(\|\cdot\|_W = max_x|\cdot/W(x)|)$

1). $x \in S$, where S is a finite set.
2). $\sum_{i=0}^n \alpha_i(x) = \infty$, $\sum_{i=0}^n \alpha_i^2(x) < \infty$.
3). $\gamma \in (0, 1)$.

Proof. The proof is in the Appendix A.3.

Lemma 2. *Consider a random iterative process* $\Delta_{n+1} = (1-\alpha_n(x))\Delta_n(x) + \alpha_n(x)F_n(x)$ *under the following conditions:*

1). $x \in S$, where S is a finite set.
2). $\sum_{i=0}^n \alpha_i(x) = \infty$, $\sum_{i=0}^n \alpha_i^2(x) < \infty$, where $\alpha_n(x)$ is a nonnegative number.
3). $\|E\{F_n(x)|P_n\}\|_W \leq \gamma(\|\Delta_n\|_W + \max_t \|\Delta_n - \Delta_{n-t}\|_W)$, and $\gamma \in (0, 1), t \in [0, K-1], K > 0$.
4). $Var\{F_n(x)|P_n\} \leq C(1 + \|\Delta_n\|_W)^2$, where C is some constant. Among the above assumptions, $P_n = \{X_n, \ldots, X_0; \alpha_n, \ldots, \alpha_0; F_{n-1}, \ldots, F_0\}$.

$\Delta_n(x)$ converges to zero with possibility one if the above conditions are satisfied.

Proof. The proof is in the Appendix A.3.

Theorem 1. *Let us define a stochastic iterative process satisfying the following conditions:* $(i \in [1, M])$

$$\begin{aligned} i'_* &= \operatorname*{argmax}_{i'} Q_n(s', a_{i'}) \\ Y_n^K &= \min(\underbrace{Q_n(s', a_{i'_*}), Q_{n-1}(s', a_{i'_*}), Q_{n-2}(s', a_{i'_*}), \ldots)}_{K} \\ Q_{n+1}(s, a_i) &= (1-\alpha_n(s, a_i))Q_n(s, a_i) + \alpha_n(s, a_i)[r + \gamma Y_n^K] \end{aligned} \tag{14}$$

1). the state and action spaces are finite.
2). $\sum_{j=0}^n \alpha_j(s, a_i) = \infty$, $\sum_{j=0}^n \alpha_j^2(s, a_i) < \infty$, and where $\alpha_j(s, a_i) \in [0, 1]$.
3). $\gamma \in (0, 1)$ and $Var\{r\} < \infty$.

The process converges to zero with possible one.

Proof. We next sketch how to apply lemma 2 to prove Theorem 1 without going into full technical details. By subtracting $Q^*(s, a_i)$ from both sides of Equ (14), we get that $F_n(s, a_i) = r + \gamma Y_n^K - Q^*(s, a_i)$ and define $\Delta_n(s, a_i) = Q_n(s, a_i) - Q^*(s, a_i)$. The maximum norm of $E\{F_n(s, a_i)\}$ satisfies the condition 3) in lemma 2:

$$\begin{aligned} \|E\{F_n(s, a_i)\}\| &= \gamma\|Y_n^K - \max_{i''} Q^*(s', a_{i''})\| \\ &\leq \gamma\|Y_n^K - Q_n(s', a_{i'_*})\| + \gamma\|Q_n(s', a_{i'_*}) - \max_{i''} Q^*(s', a_{i''})\| \\ &\leq \gamma(\max_{t,i} \|\Delta_n(s', a_i) - \Delta_{n-t}(s', a_i)\| + \max_i \|\Delta_n(s', a_i)\|) \end{aligned} \tag{15}$$

where $t \in [0, K]$. The update of $Q_{n+1}(s, a_i)$ primarily depends on the linear sum of γr and the $Var(r)$ and $Q_0(s, a)$ are bounded. Accordingly, $\|\Delta_n\|$ and the variance of $F_n(s, a_i)$ are bounded by some values. The conditions are satisfied and *successively pruned q-learning* can converge to the optimal value.

Algorithm 1 Successively Pruned Q-learning

Initialize s and $Q_0(s, a)$ for all state-action pairs
Set step size $\alpha \in (0, 1]$, discount factor $\gamma \in (0, 1]$, recall size $K \geq 1$ and $\epsilon > 0$
repeat
  Choose a from s using the $\epsilon-$greedy policy in Q
  Take action a, observe r, s' and update $Q_n(s, a)$:
  Define $i'_* = \arg\max_{i'} Q_n(s', a_{i'})$
  $Y_n^K = \min(\underbrace{Q_n(s', a_{i'_*}), Q_{n-1}(s', a_{i'_*}), Q_{n-2}(s', a_{i'_*}), \ldots)}_{K}$
  $Q_{n+1}(s, a) = (1-\alpha_n(s, a))Q_n(s, a) + \alpha_n(s, a)[r + \gamma Y_n^K]$
  $s \leftarrow s'$
until end

5 GENERALIZED TO OFF-POLICY ALGORITHMS

In this section, we show that *successively pruned q-learning* can be generalized to discrete and continuous control domain, combined with Deep Q Network (DQN) [23], Deep Deterministic Policy Gradient (DDPG) [23], Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [13], and Soft Actor-Critic (SAC) [14].

5.1 Successively pruned DQN

We can naturally combine *successively pruned q-learning* with DQN [23] to generate our deep neural network algorithm called Successively Pruned DQN (SPDQN). As we all know, DQN has two significant networks comprising online network and target network. Online network presents advanced Q value and the target network is devised to stabilize the learning process. The target network is derived from the online network but uses a delayed update mechanism that its parameters are copied every C steps according to the online network. The target network is a natural concept to record the previous Q value. Therefore, multiple target networks can be used to express $\{Q_n, Q_{n-1}, Q_{n-2}, \ldots\}$. As for the update method of multiple target network, we propose two methods: the first is to update jth target network every jC steps through online network, the second is to update jth target network every jC steps by $j-1$th target network. The target networks in the first way can be synchronous with the online network but the target networks in the second always delay a version, which decreases the convergence speed. Therefore, the first update way is more proper considering the training resources. Besides, we can view the first target network as stable replacement of the online network, so that the action selection can happen in the online network or the first target network. In summary, the algorithmic framework is as follows:

1). $i'_* = \underset{i'}{\operatorname{argmax}}\, Q(S_{t+1}, a_{i'}; \theta_t)$ or $i'_* = \underset{i'}{\operatorname{argmax}}\, Q(S_{t+1}, a_{i'}; \theta_t^-)$

2). $Y_t^{\text{SPDQN}} = R_{t+1} + \gamma \min \underbrace{(Q(S_{t+1}, a_{i'_*}; \theta_t^-), Q(S_{t+1}, a_{i'_*}; \theta_t^=), \ldots)}_{K}$

where $\{\theta_t^-, \theta_t^=, \ldots\}$ denote the delayed parameters of multiple target networks and according to the source of action selection, our method evolves into SPDQN-T using the first target network to choose action and SPDQN-O employing the online network.

5.2 Combined with off-policy actor-critic algorithms

The construction of off-policy actor-critic algorithms is diverse from DQN, which only owns several critic networks. In general, off-policy actor-critic algorithms possess several critic networks and actor networks. Owing to the existence of the actor networks, the behavior of action selection can be ignored. Our method can be employed straight to DDPG through minimizing between multiple target networks. As for TD3 and SAC, two critic online networks are utilized to reduce the overestimation on the actor-critic framework. Successively pruned method can be directly augmented to clipped double q-learning, further controlling the bias.

6 EXPERIMENTS

In this section, we conduct experiments to illustrate the effectiveness of our proposed method. Firstly, we consider windy walking task in [33] and augment random winds and rewards to challenge the Q-learning algorithm and its variants. And then, we focus on the Atari 2600 games, using the Arcade Learning Environment [5] with performance comparison of several baseline algorithms. To speed up the training procedure, we apply the asynchronous mode

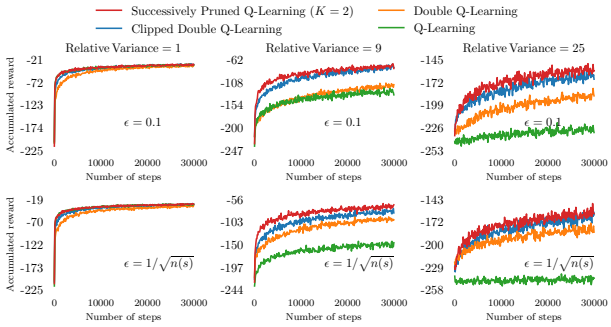

Figure 1: The stochastic rewards are sampled from -1 and 0 with equal probability, with relative variance considered as 1. (-2,1) (relative variance=9) and (-3,2) (relative variance=25) are chosen as subsequent rewards in environments. All experiments are run 30000 steps and averaged over 500 runs.

with cpu parallel samplers in the rlpyt [30], which uses many accelerated methods mentioned in [29]. Finally, to evaluate off-policy actor-critic algorithms, the suite of MuJoCo continuous control tasks [35] are examined. All detailed description of experiment settings is deferred to Appendix B.

6.1 Windy walking

An windy walking task in Example 6.5 of [33] contains the starting state and the goal state. In contrast with standard grid world, there exists upward winds that vary in intensity and location. The agent can execute 4 actions including upwards, down, left and right. Each column of wind power varies according to the number shown at the bottom of each column. $+n$ denotes that the agent is blew to move upwards n grids and $-m$ is to move down m grids. Moreover, stochastic wind power and reward are additionally augmented to enhance the randomness of the task. The stochastic wind power is mentioned in the Exercise 6.10 of [33] and means that the agent can additionally move upwards 1 grid or down 1 grid with equal possibility. The stochastic reward follows the definition in [15] and the agent receives different rewards at each step with equal probability.

We use accumulated reward to measure the performance of different algorithms in Figure 1. We mainly analyse two exploration strategies: $\epsilon = 0.1$ and $\epsilon = 1/\sqrt{n(s)}$ and set the step size $\alpha_n(s,a) = 1/n(s,a)$, which $n(s,a)$ denotes the times (s,a) has been visited. Stochastic reward $(-1, 0)$ is chosen in the initial experiments, whose variance is small. From the first column of Figure 1, we can see that Successively Pruned Q-learning ($K = 2$) and Q-learning have the almost same convergence speed with low-variance reward. Conversely, Double Q-learning and Clipped Double Q-learning converge more slowly than them due to the random update of two Q-functions, which is consistent in both two strategies. These experiments indicate that Double Q-learning is not good at processing tasks with low-variance reward and Q-learning can not tackle the tasks with high-variance reward well. Clipped Double Q-learning performs more similarly to our method with the increase of the randomness. Our Successively Pruned Q-learning ($K = 2$)

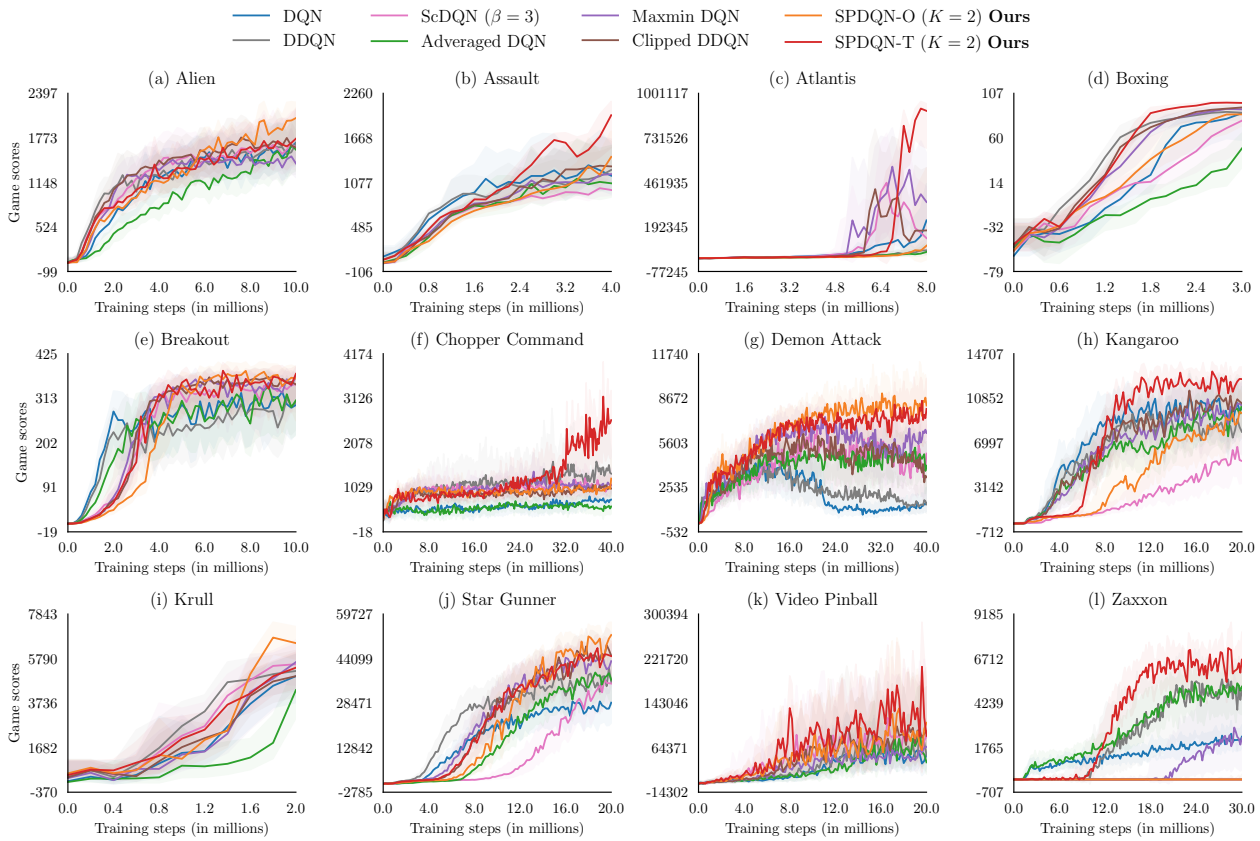

Figure 2: Learning curves for the Atari tasks. The shaded region represents three-quarters of a standard deviation of the average evaluation over 6 trials.

algorithm performs well in both variances due to its special update rule, which mainly chooses the minimum between $Q_n(s', a_{i'_*})$ and $Q_{n-1}(s', a_{i'_*})$. If the reward has low variance, $Q_n(s', a_{i'_*})$ and $Q_{n-1}(s', a_{i'_*})$ are close to each other. Hence, Successively Pruned Q-learning algorithm is more similar to Q-learning and randomly selecting between two Q-functions like Double Q-learning would waste computational resources and slow the convergence speed. Although Clipped Double Q-learning switches from two Q-functions, the min operator can reduce the variance. If the reward has high variance, single $Q_n(s', a_{i'_*})$ exists serious overestimation and it will lead to bad performance. Converting between two Q-functions can reduce the overestimation, which is the reason why Successively Pruned Q-learning, Double Q-learning and Clipped Double Q-learning work better.

6.2 Atari benchmark for DQN algorithms

To demonstrate the superiority of SPDQN-T and SPDQN-O, we compare with six variants of deep Q-networks as baselines, including DQN [23], Double DQN (DDQN) [36], Adveraged DQN (ADQN) [1], Clipped DDQN (CDDQN) [13], Maxmin DQN (MDQN) [20] and Self-correcting DQN (ScDQN) [42] in twelve atari tasks. To ensure the fairness of comparison, ensemble methods employ the parameter $N = 2$. Given the recent concerns in reproducibility, our experiments are run six times with fair evaluation metrics, and open source both learning curves and open code [1]. Our results are shown in Figure 2 exhibiting learning curves over 6 trials and Table 1 listing the average scores of final policies over 40 trials. The results show that, overall, SPDQN-O or SPDQN-T performs comparably to the baseline methods from convergence value and stability particularly in Atlantis or Demon Attack games where our methods perform stably although other algorithms drop down with the increasing of training steps. In most tasks, although our methods converge slower than others, our methods can achieve more higher game scores. As for SPDQN-O and SPDQN-T, the performance of SPDQN-T is better in most atari tasks and SPDQN-T performs not worse in several tasks where SPDQN-O surpasses all compared algorithms. This is because using the first target network to determine the action is closer to other target networks so that the training score can steadily rise. All hyper-parameters used in the Atari tasks are listed in Appendix B.

[1] https://github.com/dfasdffe/Successively-Pruned-Q-Learning/tree/master

Environment	SPDQN-T	SPDQN-O	DQN	DDQN	ScDQN	ADQN	MDQN	CDDQN
Alien	1693± 536	**1882 ± 569**	1590	1632	1696	1520	1462	1688
Assault	**1593 ± 656**	1261 ± 452	1271	1202	975	1217	1245	1349
Atlantis	**767361±297179**	158562 ± 298223	175927	55601	297483	40879	428610	262630
Boxing	**95 ± 5**	72±22	69	85	54	26	87	87
Breakout	**345 ± 90**	334 ± 107	286	269	317	291	329	338
Chopper Command	**2471 ±1452**	1088± 435	736	1592	1059	589	1121	1076
Demon Attack	7570 ± 3170	**8155 ± 3496**	1439	1737	4476	4424	5848	3940
Kangaroo	**11870 ±2876**	8809 ± 3577	9475	8203	6025	9282	9638	10256
Krull	5082±1362	**6613±1495**	4675	5148	5501	3225	5099	4876
Star Gunner	43626 ± 11903	**48998 ± 12599**	27463	35696	33168	36954	42017	44838
Video Pinball	**132727 ± 141796**	113811 ± 136329	64878	84229	106464	80875	61636	109647
Zaxxon	**6472 ± 2422**	32 ± 237	2272	4325	4	4927	2836	41

Table 1: Average scores over 40 trials of final policies. Maximum value for each task is bolded. ± corresponds to a single standard deviation over trials. The deviations of other baseline algorithms are shown in Appendix C.

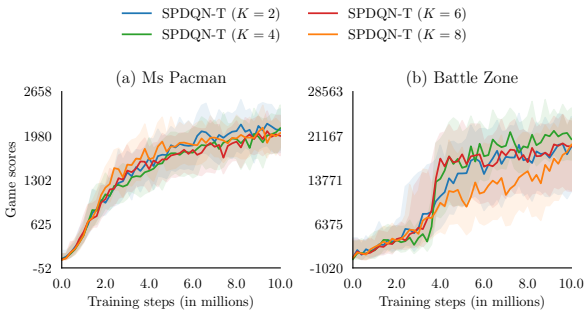

Figure 3: Training curves for different K over 6 runs and the shaded region represents a standard deviation.

Selection of the parameter K. K presents the number of joining in minimum. We choose Ms Pacman and Battle Zone to demonstrate the effect of K in Figure 3. The two tasks represent the different situations respectively. Some tasks are not sensitive to the parameter K like Ms Pacman and in other tasks such as Battle Zone, larger K decreases the convergence speed, but maybe can jump out of the sub-optimal value and attain higher value, which demonstrated by the gradient of 10M steps on $K = 8$.

6.3 Mujoco environments for off-policy actor-critic algorithms

The aim of experimental evaluations is to analyse the benefits the off-policy actor-critic algorithms can gain from the successively pruned tool. Off-policy methods do not presume that the samples are derived from the current trained policy. In fact, this denotes the samples on the replay buffer can be reused multiple times through back-propagations. To equitably analyse the effect of successively pruned method, we employ the open code stable-baselines3 [2] and the hyper-parameter setting basically obeys the default, listed in Appendix B. The successively pruned tool is augmented to three important off-policy algorithms, comprising Deep Deterministic Policy Gradient (DDPG) [22], Twin Delayed Deep Deterministic Policy Gradient (TD3) [13] and Soft Actor-Critic (SAC) [14]. A range of challenging continuous control tasks from the OpenAI gym benchmark suite [6] are chosen as the evaluation experiments.

Table 2 reveals average game scores of evaluation rollouts during training for DDPG, TD3, TD3-CDQ (TD3 removing clipped double q-learning section), SAC and their variants with successively pruned tool, in which the numbers respectively correspond to the mean and the standard deviation over 6 trials of 1 million steps. Table 3 shows the benefits that different algorithms gain from successively pruned tool, that is, we record average game scores over 10 trials of final policies on Table 2 and consider the ratio of specific algorithm with successively pruned tool to itself minus one as the improvement. Particularly, we determine original TD3-CDQ without any method concerning reducing the overestimation as the comparison object for TD3 and TD3-SP. The results show that our method added to the off-policy algorithms has ameliorated performance in most environments. In general, the benefit that SAC employs successively pruned tool is less than others. This is because SAC uses a Update-To-Data (UTD) [8] of 1 and other off-policy algorithms employs a UTD $\ll 1$. Besides, the learning rate used in the Adam optimizer [18] of SAC is 0.0003 and others are 0.001. These factors ensure the stability of training process and little gradient value and many updates are hard to benefit from the tool of reducing the overestimation bias.

Deterministic policy gradient algorithm [28] proposes a challenging optimization situation of the policy parameterization, meaning that the general benefits of over-parameterized networks do not

[2] https://github.com/DLR-RM/stable-baselines3

Environment	DDPG	DDPG-SP ($K = 4$)	TD3-CDQ	TD3	TD3-SP ($K = 4$)	SAC	SAC-SP ($K = 4$)
Ant	**513 ± 483**	465 ± 440	338 ± 686	3327 ± 691	**4469 ± 1666**	4734 ±671	**5428 ± 690**
HalfCheetah	4117 ±188	**5163 ±459**	3329± 223	6779 ± 295	**8978 ± 195**	8810 ± 296	**10293 ± 306**
Hopper	1398 ± 1005	**1899± 1029**	1553 ±1185	1775± 1590	**3508± 423**	3280± 212	**3469± 242**
Humanoid	88 ± 22	**302 ± 84**	157±131	91 ± 28	**400 ± 120**	5145 ± 436	**5540 ± 568**
Pusher	**-32 ± 4**	-37 ± 17	**-31 ± 6**	-50 ±7	-48 ± 7	**-23 ± 2**	**-23 ± 2**
Walker2d	2807 ± 1462	**3478 ± 1543**	2464 ± 1221	3991 ± 795	**4288 ± 468**	4536 ± 240	**4762 ± 615**

Table 2: Average scores over 10 trials of 1 million steps for the MuJoco continuous control tasks. ± corresponds to a single standard deviation over trials. Maximum value for each algorithm is bolded. Corresponding learning curves defer to Appendix C.

Environment	DDPG-SP ($K = 4$)	TD3	TD3-SP ($K = 4$)	SAC-SP ($K = 4$)
Ant	-9.36%	884.31%	1222.19%	14.66%
HalfCheetah	25.41%	103.63%	169.69%	16.83%
Hopper	35.84%	14.29 %	125.89%	5.76%
Humanoid	243.18%	-42.04%	154.78 %	7.68%
Pusher	-13.51%	-61.29%	-54.84%	0%
Walker2d	23.90%	61.97%	74.03%	4.98%

Table 3: The benefits that off-policy actor-critic algorithms gain from tools reducing the overestimation.

totally overcome issues concerning local optima, which is part of the reason why different runs of the same algorithm can produce drastically different solutions in DDPG and TD3. This characteristic leads to the sensitivity to different Q values and our minor Q values maybe can conduct the policy to get out from many local optima and reach a better point. In most environments, we can observe that DDPG gains significant benefits from our method. However, the structure of DDPG makes it hard to benefit from the environments with large observation shape such as Ant and Pusher, in which DDPG performs more like a kind of stochastic behavior. TD3-CDQ transforms the network structure and augments target policy smoothing regularization and delayed policy updates, which help the methods of reducing the overestimation bias ameliorate the algorithm performance. TD3 with clipped double q-learning can stabilize the Q value and produces a marked effect, especially in Ant and HalfCheetah environments and this is reflected empirically that TD3-SP ($K = 4$) with successively pruned algorithm incorporated into clipped double q-learning can further alleviate the overestimation bias and enhance the performance. The same principle applies to other variants relating to Q-learning.

7 CONCLUSION

We mainly have presented a novel algorithm called *successively pruned q-learning* to alleviate the positive bias during the update in this paper. Firstly, we analyse that the accumulated error on Q-learning is bounded by the historical values and then propose to utilize the preceding values to control the sequential error. After that we demonstrate that our successively pruned estimator not only balances between the overestimation and the underestimation, but also can enjoy the minimum bias against the true value under appropriate setting. Moreover, our algorithm can be naturally extended to deep learning domain. In tabular setting, we show that our algorithm simultaneously owns the strengths of Q-learning and Double Q-learning so that it dose well in the tasks with low-variance and high-variance rewards. Our Successively Pruned DQN outperforms DQN and other variants in most of the Atari games and some classical off-policy actor-critic algorithms can gain significant benefits from our method.

In our future directions, we expect to advance our model from the following items: 1) Incorporate our method with other variants of Q-learning to explore the best way to reduce the overestimation bias. 2) No matter in DQN or other off-policy actor-critic algorithms, the variance of our method does not give an advantage against others and this requires us to devise more accurate model. 3) Our algorithm can be enlarged to multi-agent reinforcement learning and combined with value-based methods like Value-Decomposition Networks [32] and QMIX [26] et al.

ACKNOWLEDGMENTS

This work is supported in part by Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103), the National Key R&D Program of China (2021ZD0113503) and Ningbo Science and Technology Bureau, Technology and Equipment of Intelligent and All-terrain Unmanned Aerial System for Magmatic Exploration. No.2020Z073. This project was also funded by the National Natural Science Foundation of China 82090052.

REFERENCES

[1] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*. PMLR, 176–185.

[2] Masanao Aoki. 2016. *Optimization of stochastic systems: topics in discrete-time systems*. Elsevier.

[3] K Johan Astrom. 1987. Adaptive feedback control. *Proc. IEEE* 75, 2 (1987), 185–217.

[4] Mohammad Gheshlaghi Azar, Remi Munos, Mohammad Ghavamzadeh, and Hilbert Kappen. 2011. Speedy Q-learning. In *Advances in neural information processing systems*.

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[7] Gang Chen. 2020. Decorrelated double q-learning. *arXiv preprint arXiv:2006.06956* (2020).

[8] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. 2021. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982* (2021).

[9] Carlo D'Eramo, Andrea Cini, Alessandro Nuara, Matteo Pirotta, Cesare Alippi, Jan Peters, and Marcello Restelli. 2021. Gaussian approximation for bias reduction in Q-learning. *The Journal of Machine Learning Research* 22, 1 (2021), 12690–12740.

[10] Adithya M Devraj and Sean P Meyn. 2017. Fastest convergence for Q-learning. *arXiv preprint arXiv:1707.03770* (2017).

[11] Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. 2016. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*. PMLR, 1032–1040.

[12] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6 (2005).

[13] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[15] Hado Hasselt. 2010. Double Q-learning. *Advances in neural information processing systems* 23 (2010).

[16] Tommi Jaakkola, Michael Jordan, and Satinder Singh. 1993. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems* 6 (1993).

[17] Haobo Jiang, Jin Xie, and Jian Yang. 2021. Action candidate based clipped double q-learning for discrete and continuous action tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7979–7986.

[18] D Kinga, Jimmy Ba Adam, et al. 2015. A method for stochastic optimization. In *International conference on learning representations (ICLR)*, Vol. 5. San Diego, California;, 6.

[19] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. 2020. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*. PMLR, 5556–5566.

[20] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. 2020. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487* (2020).

[21] Donghun Lee, Boris Defourny, and Warren B Powell. 2013. Bias-corrected q-learning to control max-operator bias in q-learning. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 93–99.

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[24] Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. 2021. Ensemble bootstrapping for Q-Learning. In *International Conference on Machine Learning*. PMLR, 8454–8463.

[25] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. 2018. Toward end-to-end control for UAV autonomous landing via deep reinforcement learning. In *2018 International conference on unmanned aircraft systems (ICUAS)*. IEEE, 115–123.

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[27] Zhizhou Ren, Guangxiang Zhu, Hao Hu, Beining Han, Jianglun Chen, and Chongjie Zhang. 2021. On the estimation bias in double Q-learning. *Advances in Neural Information Processing Systems* 34 (2021), 10246–10259.

[28] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. Pmlr, 387–395.

[29] Adam Stooke and Pieter Abbeel. 2018. Accelerated methods for deep reinforcement learning. *arXiv preprint arXiv:1803.02811* (2018).

[30] Adam Stooke and Pieter Abbeel. 2019. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500* (2019).

[31] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. 2006. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*. 881–888.

[32] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[33] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[34] Sebastian Thrun and Anton Schwartz. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*, Vol. 255. Hillsdale, NJ, 263.

[35] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[36] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[37] Hang Wang, Sen Lin, and Junshan Zhang. 2021. Adaptive ensemble q-learning: Minimizing estimation bias via error feedback. *Advances in Neural Information Processing Systems* 34 (2021), 24778–24790.

[38] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.

[39] Wentao Weng, Harsh Gupta, Niao He, Lei Ying, and R Srikant. 2020. The mean-squared error of double Q-learning. *Advances in Neural Information Processing Systems* 33 (2020), 6815–6826.

[40] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. 2015. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791* (2015).

[41] Zongzhang Zhang, Zhiyuan Pan, and Mykel J Kochenderfer. 2017. Weighted double Q-learning.. In *IJCAI*. 3455–3461.

[42] Rong Zhu and Mattia Rigotti. 2021. Self-correcting q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11185–11192.