Weighted Double Q-learning

Zongzhang Zhang
Soochow University
Suzhou, Jiangsu 215006 China
zzzhang@suda.edu.cn

Zhiyuan Pan
Soochow University
Suzhou, Jiangsu 215006 China
owenpzy@gmail.com

Mykel J. Kochenderfer
Stanford University
Stanford, CA 94305 USA
mykel@stanford.edu

Abstract

Q-learning is a popular reinforcement learning algorithm, but it can perform poorly in stochastic environments due to overestimating action values. Overestimation is due to the use of a single estimator that uses the maximum action value as an approximation for the maximum expected action value. To avoid overestimation in Q-learning, the double Q-learning algorithm was recently proposed, which uses the double estimator method. It uses two estimators from independent sets of experiences, with one estimator determining the maximizing action and the other providing the estimate of its value. Double Q-learning sometimes underestimates the action values. This paper introduces a weighted double Q-learning algorithm, which is based on the construction of the weighted double estimator, with the goal of balancing between the overestimation in the single estimator and the underestimation in the double estimator. Empirically, the new algorithm is shown to perform well on several MDP problems.

1 Introduction

Sequential decision problems under uncertainty are often framed as Markov decision processes (MDPs) [Kochenderfer, 2015; Bertsekas, 2007]. Reinforcement learning is concerned with finding an optimal decision making strategy in problems where the transition model and rewards are not initially known [Littman, 2015; Wiering and van Otterlo, 2012; Sutton and Barto, 1998]. Some reinforcement learning algorithms involve building explicit models of the transitions and rewards [Brafman and Tennenholtz, 2001], but other "model-free" algorithms learn the values of different actions directly.

One of the most popular model-free algorithms is Q-learning [Watkins, 1989]. The original Q-learning algorithm inspired several improvements, such as delayed Q-learning [Strehl *et al.*, 2006], phased Q-learning [Kearns and Singh, 1999], fitted Q-iteration [Ernst *et al.*, 2005], bias-corrected Q-learning [Lee and Powell, 2012; Lee *et al.*, 2013], and weighted Q-learning [D'Eramo *et al.*, 2016].

This paper focuses on an enhancement known as double Q-learning [van Hasselt, 2010; 2011], a variant designed to avoid the positive maximization bias when learning the action values. The algorithm has recently been generalized from the discrete setting to use deep neural networks [LeCun *et al.*, 2015] as a way to approximate the action values in high dimensional spaces [van Hasselt *et al.*, 2016]. Double Q-learning, however, can lead to a bias that results in underestimating action values.

The main contribution of this paper is the introduction of the weighted double Q-learning algorithm, which is based on the construction of the weighted double estimator, with the goal of balancing between the overestimation in the single estimator and underestimation in the double estimator. We present empirical results of estimators of the maximum expected value on three groups of multi-arm bandit problems, and compare Q-learning and its variants in terms of the action-value estimate and policy quality on MDP problems.

2 Background

The MDP framework can be applied whenever we have an agent taking a sequence of actions in a system described as a tuple (S, A, T, R, γ), where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \to [0, 1]$ is a state-transition model, where $T(s, a, s')$ gives the probability distribution over state s' after the agent executes action a in state s, $R : S \times A \to \mathbb{R}$ is a reward function, where $R(s, a)$ gives the reward obtained by the agent after executing action a in state s, and $\gamma \in [0, 1)$ is a discount factor that trades off the importance of immediate and delayed rewards.

An MDP policy is a mapping from S to A, denoted by $\pi : S \to A$. The goal of solving an MDP is to find an optimal policy π^* that maximizes $V^\pi : S \to \mathbb{R}$, the value of a state s under policy π, defined as

$$E_\pi \Big\{ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s \Big\}. \tag{1}$$

A similar state-action value function is $Q^\pi : S \times A \to \mathbb{R}$, where $Q^\pi(s, a)$ is the value of starting in state s, taking action a, and then continuing with the policy π. The optimal state-value function $Q^*(s, a)$ in the MDP framework satisfies the Bellman optimality equation [Bellman, 1957]:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \max_{a'} Q^*(s', a'). \tag{2}$$

We can use Q^* to define $\pi^*(s) \in \arg\max_{a \in A} Q^*(s, a)$.

3 Estimators of the Maximum Expected Value

This section considers the problem of finding an approximation for $\max_i E\{X_i\}$, the maximum expected value of the set of N random variables $X = \{X_1, X_2, \dots, X_N\}$. We first describe two existing methods, the single estimator and the double estimator, and then introduce our new weighted double estimator method.

3.1 Single Estimator

Let $\mu = \{\mu_1, \mu_2, \dots, \mu_N\}$ be a set of unbiased estimators such that $E\{\mu_i\} = E\{X_i\}$, for all i. Assume that $D = \cup_{i=1}^{N} D_i$ is a set of samples, where D_i is the subset containing at least one sample for the variable X_i, and the samples in D_i are i.i.d. The single estimator method uses the value $\max_i \mu_i(D)$ as an estimator of $\max_i E\{X_i\}$, where $\mu_i(D) = \frac{1}{|D_i|}\sum_{d\in D_i} d$ is an unbiased estimator for the value of $E\{X_i\}$. However, $\max_i E\{\mu_i(D)\} \geq \max_i E\{X_i\}$, and the inequality is strict if and only if $P(j \notin \arg\max_i \mu_i(D)) > 0$ for any $j \in \arg\max_i E\{X_i\}$ [van Hasselt, 2010]. This implies that there will be a positive maximization bias if we use the maximum of the estimates as an estimate of the maximum of the true values. Such a biased estimate can happen when all variables in X are i.i.d. The overestimation in the single estimator method is due to the same samples are both used to determine the maximizing action and to estimate its value.

3.2 Double Estimator

To avoid maximization bias in the single estimator, Hasselt proposed the double estimator approach [van Hasselt, 2010]. One of the key ideas is to divide the sample set D into two disjoint subsets, D^U and D^V. Let $\mu^U = \{\mu_1^U, \mu_2^U, \dots, \mu_N^U\}$ and $\mu^V = \{\mu_1^V, \mu_2^V, \dots, \mu_N^V\}$ be two sets of unbiased estimators such that $E\{\mu_i^U\} = E\{\mu_i^V\} = E\{X_i\}$ for all i. The two sample subsets are used to learn two independent estimates, $\mu_i^U(D) = \frac{1}{|D_i^U|}\sum_{d\in D_i^U} d$ and $\mu_i^V(D) = \frac{1}{|D_i^V|}\sum_{d\in D_i^V} d$, each an estimate of the true value $E\{X_i\}$ for all i. The value $\mu_i^U(D)$ is used to determine the maximizing action $a^* \in \arg\max_i \mu_i^U(D)$, and the other is used to provide the estimate of its value, $\mu_{a^*}^V(D)$. This estimate will then be unbiased in the sense that $E\{\mu_{a^*}^V(D)\} = E\{X_{a^*}\}$. However, since $E\{X_{a^*}\} \leq \max_i E\{X_i\}$, we have $E\{\mu_{a^*}^V(D)\} \leq \max_i E\{X_i\}$. The inequality is strict if and only if $P(a^* \notin \arg\max_i E\{X_i\}) > 0$ [van Hasselt, 2010]. This implies that there will be a negative bias if we use $\mu_{a^*}^V$ as an estimate of the maximum of the true values when the variables have different expected values and overlapped distributions. When the variables are i.i.d., the double estimator is unbiased because all expected values are equal and $P(a^* \in \arg\max_i E\{X_i\}) = 1$.

3.3 Weighted Double Estimator

Our weighted double estimator method provides a way of constructing a set of estimators that balances the overestimation of the single estimator and the underestimation of the double estimator. Mathematically, we write it as follows:

$$\mu^{WDE}(D) = \beta\mu_{a^*}^U(D) + (1-\beta)\mu_{a^*}^V(D), \tag{3}$$

Algorithm 1 Q-learning

```
1: Initialize $Q$, $s$
2: loop
3:     Choose action $a$ from state $s$ based on $Q$ and some exploration strategy (e.g., $\epsilon$-greedy)
4:     Take action $a$, observe $r$, $s'$
5:     $a^* \leftarrow \arg\max_a Q(s', a)$
6:     $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
7:     $Q(s, a) \leftarrow Q(s, a) + \alpha(s, a)\delta$
8:     $s \leftarrow s'$
```

where $\beta \in [0, 1]$ and $a^* \in \arg\max_i \mu_i^U(D)$. Thus, $\mu^{WDE}(D)$ equals the result of the single estimator when $\beta = 1$, and the double estimator when $\beta = 0$.

Now we consider how to construct the function β. Assume that the variable X_i follows the distribution H_i, i.e., $X_i \sim H_i$. Denote the Kullback-Leibler divergence between the two distributions H_i and H_j as $KL(H_i \parallel H_j)$. When $\max_{i,j} KL(H_i \parallel H_j) = 0$, the variables X_i in X are i.i.d. To make $\mu^{WDE}(D)$ be an unbiased estimate for $\max_i E\{X_i\}$, β should be set to 0. Similarly, when $\max_{i,j} KL(H_i \parallel H_j)$ is small, we will want to also set β to a small value. We hypothesize that $KL(H_{a^*} \parallel H_{a_L})$, where $a_L \in \arg\min_i \mu_i^U(D)$, can serve as an approximation to $\max_{i,j} KL(H_i \parallel H_j)$, because X_{a^*} and X_{a_L} are the two variables with the biggest difference in terms of the expected values in the sample subset D^U. Since the distributions of variables are unavailable, we further use $|E\{X_{a^*}\} - E\{X_{a_L}\}|$ to approximate $KL(H_{a^*} \parallel H_{a_L})$. Since $|\mu_{a^*}^V(D) - \mu_{a_L}^V(D)|$ is an unbiased estimator of $|E\{X_{a^*}\} - E\{X_{a_L}\}|$, we define β as follows:

$$\beta(D, c) = \frac{|\mu_{a^*}^V(D) - \mu_{a_L}^V(D)|}{c + |\mu_{a^*}^V(D) - \mu_{a_L}^V(D)|}, \tag{4}$$

where $c \geq 0$. Because there exists one $\beta^* \in [0, 1]$ such that $\beta^* E\{\mu_{a^*}^U(D)\} + (1-\beta^*)E\{\mu_{a^*}^V(D)\} = \max_i E\{X_i\}$, and, for any $\beta^* \in [0, 1]$, there is a corresponding $c^* \in [0, +\infty)$ such that $\beta^* = E\{\beta(D, c^*)\}$, we can conclude that there always exists one $c^* \in [0, +\infty)$ such that $\mu^{WDE}(D, c^*)$ is an unbiased estimator of $\max_i E\{X_i\}$. Note that even if c is a constant, Eq. (4) can ensure that the more similar the distributions that variables follow, the closer the gap between $\beta(D, c)$ and 0 is. Tuning parameter β directly cannot achieve this. Selecting c is discussed in a later section.

4 Q-learning and Its Variants

This section first describes Q-learning and double Q-learning, and then presents the weighted double Q-learning algorithm.

4.1 Q-learning

Q-learning is outlined in Algorithm 1. The key idea is to apply incremental estimation to the Bellman optimality equation. Instead of using T and R, it uses the observed immediate reward r and next state s' to obtain the following update rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(s, a)[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], \tag{5}$$

Algorithm 2 Double Q-learning

```
 1: Initialize $Q^U$, $Q^V$, $s$
 2: loop
 3:     Choose $a$ from $s$ based on $Q^U$ and $Q^V$ (e.g., $\epsilon$-greedy in $Q^U + Q^V$)
 4:     Take action $a$, observe $r$, $s'$
 5:     Choose (e.g. random) whether to update $Q^U$ or $Q^V$
 6:     if chose to update $Q^U$ then
 7:         $a^* \leftarrow \arg\max_a Q^U(s', a)$
 8:         $\delta \leftarrow r + \gamma Q^V(s', a^*) - Q^U(s, a)$
 9:         $Q^U(s, a) \leftarrow Q^U(s, a) + \alpha^U(s, a)\delta$
10:     else if chose to update $Q^V$ then
11:         $a^* \leftarrow \arg\max_a Q^V(s', a)$
12:         $\delta \leftarrow r + \gamma Q^U(s', a^*) - Q^V(s, a)$
13:         $Q^V(s, a) \leftarrow Q^V(s, a) + \alpha^V(s, a)\delta$
14:     $s \leftarrow s'$
```

where $\alpha(s, a) \in [0, 1]$ is the learning rate. Q-learning can be interpreted as using the single estimator method to estimate the maximum expected value of $Q(s', a')$, $\max_{a'} E\{Q(s', a')\}$. Here, $\max_{a'} Q(s', a')$ is the single estimator. Since $\max_{a'} Q(s', a')$ is an unbiased sample drawn from a distribution with mean $E\{\max_{a'} Q(s', a')\}$, and $E\{\max_{a'} Q(s', a')\} \geq \max_{a'} E\{Q(s', a')\}$, the estimator $\max_{a'} Q(s', a')$ has a positive bias. Hence, Q-learning can suffer from overestimation of action values.

4.2 Double Q-learning

Double Q-learning, as shown in Algorithm 2, uses the double estimator method to estimate the maximum expected value of the Q function for the next state, $\max_{a'} E\{Q(s', a')\}$. It stores two Q functions, Q^U and Q^V, and uses two separate subsets of experience samples to learn them. The action selected to execute in line 3 is calculated based on the average of the two Q values for each action and the ϵ-greedy exploration strategy. In lines 5 to 13, with the same probability, each Q function is updated using a value from the other Q function for the next state. In line 7, the action a^* is the maximizing action in state s' based on the value function Q^U. However, in line 8, as opposed to Q-learning, the value $Q^V(s', a^*)$ but not the value $Q^U(s', a^*)$ is used to update Q^U. Since Q^V was updated with a different set of experience samples, $Q^V(s', a^*)$ is an unbiased estimate for the value of the action a^* in the sense that $E\{Q^V(s', a^*)\} = E\{Q(s', a^*)\}$. Similarly, the value of $Q^U(s', a^*)$ in line 12 is also unbiased. However, since $E\{Q^V(s', a^*)\} \leq \max_a E\{Q^V(s', a)\}$ and $E\{Q^U(s', a^*)\} \leq \max_a E\{Q^U(s', a)\}$, both estimators, $Q^V(s', a^*)$ and $Q^U(s', a^*)$, sometimes have negative biases. This results in double Q-learning underestimating action values in some stochastic environments.

4.3 Weighted Double Q-learning

Weighted double Q-learning, as outlined in Algorithm 3, combines Q-learning and double Q-learning. The key difference between Algorithm 2 and Algorithm 3 is that Algorithm 3 uses lines 8 to 10 to replace line 8, and uses lines 14 to 16 to replace line 12 in Algorithm 2. These changes make weighted double Q-learning use different δ values in updating both Q^U and Q^V. We denote

Algorithm 3 Weighted Double Q-learning

```
 1: Initialize $Q^U$, $Q^V$, $s$
 2: loop
 3:     Choose $a$ from $s$ based on $Q^U$ and $Q^V$ (e.g., $\epsilon$-greedy in $Q^U + Q^V$)
 4:     Take action $a$, observe $r$, $s'$
 5:     Choose (e.g. random) whether to update $Q^U$ or $Q^V$
 6:     if chose to update $Q^U$ then
 7:         $a^* \leftarrow \arg\max_a Q^U(s', a)$
 8:         $a_L \leftarrow \arg\min_a Q^U(s', a)$
 9:         $\beta^U \leftarrow \frac{|Q^V(s',a^*) - Q^V(s',a_L)|}{c + |Q^V(s',a^*) - Q^V(s',a_L)|}$
10:         $\delta \leftarrow r + \gamma[\beta^U Q^U(s', a^*) + (1 - \beta^U) Q^V(s', a^*)] - Q^U(s, a)$
11:         $Q^U(s, a) \leftarrow Q^U(s, a) + \alpha^U(s, a)\delta$
12:     else if chose to update $Q^V$ then
13:         $a^* \leftarrow \arg\max_a Q^V(s', a)$
14:         $a_L \leftarrow \arg\min_a Q^V(s', a)$
15:         $\beta^V \leftarrow \frac{|Q^U(s',a^*) - Q^U(s',a_L)|}{c + |Q^U(s',a^*) - Q^U(s',a_L)|}$
16:         $\delta \leftarrow r + \gamma[\beta^V Q^V(s', a^*) + (1 - \beta^V) Q^U(s', a^*)] - Q^V(s, a)$
17:         $Q^V(s, a) \leftarrow Q^V(s, a) + \alpha^V(s, a)\delta$
18:     $s \leftarrow s'$
```

$$Q^{U,WDE}(s', a^*) = \beta^U Q^U(s', a^*) + (1 - \beta^U) Q^V(s', a^*), \tag{6}$$

where

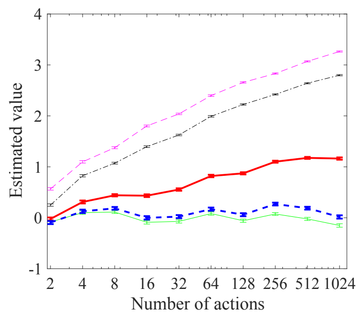
$$\beta^U = \frac{|Q^V(s', a^*) - Q^V(s', a_L)|}{c + |Q^V(s', a^*) - Q^V(s', a_L)|}. \tag{7}$$

Equations (6) and (7) are the corresponding forms of Eqs. (3) and (4) in the MDP setting, respectively. From Eq. (6) we see that weighted double Q-learning uses a linear combination of both $Q^U(s', a^*)$ and $Q^V(s', a^*)$. Thus, $Q^{U,WDE}$ represents a trade-off between the overestimation of Q-learning and the underestimation of double Q-learning. A similar update is used for Q^V, using a^*, Q^U, and Q^V.

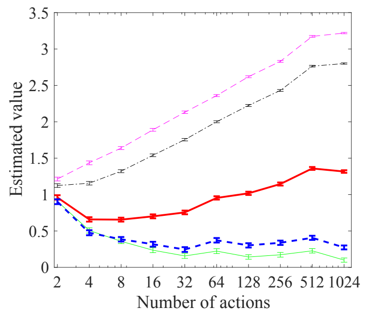
By using the proof techniques in the convergence in the limit of double Q-learning [van Hasselt, 2010], we can also prove that weighted double Q-learning converges to the optimal policy. Intuitively, this is what one would expect based on the following two observations: (1) both Q-learning and double Q-learning converge to the optimal function Q^* under similar conditions; and (2) weighted double Q-learning spans a spectrum that has the Q-learning algorithm at one end and the double Q-learning algorithm at the other.

5 Experiments

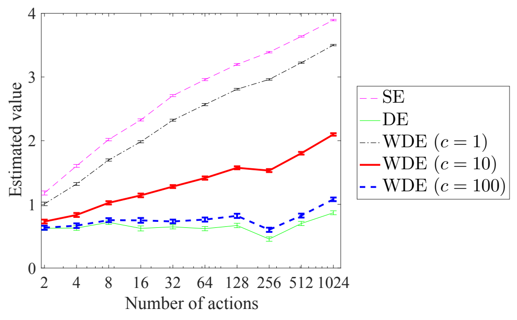
In this section, we first present empirical results of estimators of the maximum expected value on three groups of multi-arm bandit problems. Then, we present empirical comparisons of Q-learning and its variants in terms of the action-value estimate and policy quality on the game of roulette, four MDP problems modified from a 3×3 grid world problem [van Hasselt, 2010], and six intruder monitoring problems.



(a) Results on G1



(b) Results on G2



(c) Results on G3

Figure 1: Estimated values and standard errors of different estimators for maximum expected values on three groups of multi-arm bandit problems, G1, G2, and G3. SE: single estimator, DE: double estimator, WDE: weighted double estimator.

In some domains, we also compared the performance of the bias-corrected Q-learning algorithm and the weighted Q-learning algorithm, two recent proposed algorithms that address the overestimation issue for Q-learning. Bias-corrected Q-learning asymptotically cancels the max-operator bias in Q-learning by subtracting to each Q-value a bias correction term that depends on the standard deviation of the reward and on the number of actions [Lee *et al.*, 2013]. Weighted Q-learning estimates the maximum action values by a weighted estimator that computes a weighted mean of all the sample means [D'Eramo *et al.*, 2016].

Algorithms with a polynomial learning rate, $\alpha_t(s,a) = 1/n_t(s,a)^m$ with $m = 0.8$, was shown to have better performance than ones with a linear learning rate, $\alpha_t(s,a) = 1/n_t(s,a)$ [van Hasselt, 2010]. This paper focuses on empirical results of Q-learning, biased-corrected Q-learning and weighted Q-learning with parameter $m = 0.8$, and both double Q-learning and weighted double Q-learning with parameters $m^U = 0.8$ and $m^V = 0.8$ in the two learning rates $\alpha_t^U(s,a) = [1/n_t^U(s,a)]^{m^U}$ and $\alpha_t^V(s,a) = [1/n_t^V(s,a)]^{m^V}$, where the variables $n_t^U(s,a)$ and $n_t^V(s,a)$ store the number of updates in $Q^U(s,a)$ and $Q^V(s,a)$, respectively. The action-selection strategy in all algorithms was ϵ-greedy with $\epsilon(s) = 1/n_t(s)^{0.5}$, where $n_t(s)$ is the number of times state s has been visited.

5.1 Multi-arm Bandits

Our experiments are conducted on three groups of multi-arm bandit problems: (G1) $E\{X_i\} = 0$, for $i \in \{1,2,\ldots,N\}$; (G2) $E\{X_1\} = 1$ and $E\{X_i\} = 0$ for $i \in \{2,3,\ldots,N\}$; and (G3) $E\{X_i\} = \frac{i}{N}$, for $i \in \{1,2,\ldots,N\}$. Here, $E\{X_i\}$ represents the expected reward of selecting the ith action, and N is the number of actions. Thus, $\max_i E\{X_i\} = 0$ in G1, and $\max_i E\{X_i\} = 1$ in G2 and G3.

In each N-arm bandit experiment, we repeated the following procedure 100 times: generate μ_i from $\mathcal{N}(E\{X_i\}, 1)$ for all i and then generate 1000 samples d_i from $\mathcal{N}(\mu_i, 1)$. Thus, $|D_i| = 1000$. The single estimator (SE) method computes $\mu_i(D) = \frac{1}{|D_i|}\sum_{d \in D_i} d$, and then uses the average of $\max_i \mu_i(D)$ in the 100 repetitions as the single estimator of $\max_i E\{X_i\}$. The double estimator (DE) method and the weighted double estimator (WDE) method first divide D_i into two independent sample subsets, D_i^U and D_i^V, with $|D_i^U| = |D_i^V| = 500$. Then, the DE method computes $\mu_{a^*}^V(D) = \frac{1}{|D_{a^*}^V|}\sum_{d \in D_{a^*}^V} d$, where $a^* \in \arg\max_i \mu_i^U(D)$ as the double estimator; the WDE method also computes $\mu_{a^*}^V(D)$ as in the DE method, but uses the weighted average $\beta(D,c)\mu_{a^*}^U(D) + (1-\beta(D,c))\mu_{a^*}^V(D)$ in the 100 repetitions as the weighted double estimator.

Figure 1 shows empirical results for the estimated values and standard errors of different estimators for $\max_i E\{X_i\}$ on the three groups of multi-arm bandit problems. For the weighted double estimator method, we report the results with the input parameter $c = 1, 10, 100$. From this figure, we see that on all domains, as N increases from 2 to 1024, the degree of overestimation in the single estimator increases. In addition, the larger the value of c, the closer it is to the double estimator; the closer the value of c is to 0, the closer it is to the single estimator.

This figure shows that there is no fixed best value of c, even with problems with a fixed number of actions. For example, when $N = 128$ on the G1 domain, the best c is $+\infty$; on G2 the best c is around 10; and on G3 the best c is larger than 10. Although there is always a $c^* \in [0, +\infty)$ such that $\mu^{WDE}(D, c^*)$ is an unbiased estimator of $\max_i E\{X_i\}$ on each multi-arm bandit problem, the value of c^* appears to be different among problems with different characteristics. Hence, we want to set c adaptively based on the characteristics of the problem.

We observed that the number of actions clearly influences the degree of overestimations in the single estimator; however, a similar effect does not appear with the double estimator. As a result, the estimation errors in the weighted double estimator are affected by the number of actions. The empirical relationship between c in WDQ and N appears log-linear. This suggests to us that $c \propto \log_2 N$.

Table 1 shows the maximum difference between the expected value of the best action and the expected values of other actions, denoted $\Delta = \frac{E\{X_{i^*}\} - \max_{j \neq i^*} E\{X_j\}}{E\{X_{i^*}\} - \min_i E\{X_i\}}$ where $E\{X_{i^*}\} = \max_i E\{X_i\}$ (assume that $\Delta = 0$ when $E\{X_{i^*}\} - \min_i E\{X_i\} = 0$). As Δ becomes small, the double estimator tends to decrease the error caused by its underestimation, as opposed to the single estimator, which suggests setting $c \propto \frac{1}{\Delta}$.

Table 1: Estimation biases according to the single estimator and the double estimator on three groups of multi-arm bandit problems with 128 actions.

Group ($\rightarrow$)	G1	G3	G2
Difference (Δ)	0	$1/128$	1
Single estimator	2.66	2.19	1.62
Double estimator	-0.06	-0.33	-0.86

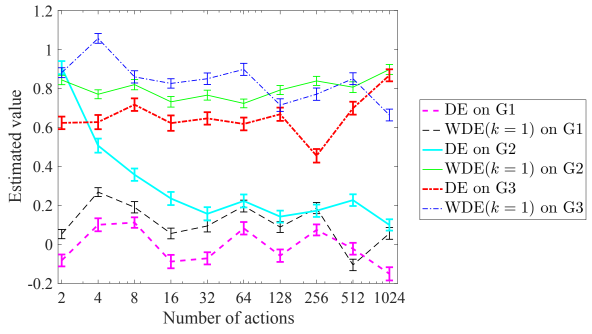


Figure 2: Comparison of WDE($k = 1$) and DE in terms of estimated values and standard errors of on three groups of multi-arm bandit problems, G1, G2, and G3.

The domain G_2 is an extreme case where only one action has the highest expected reward and all other actions have the lowest expected reward. In this case, DE has the largest estimation error, and the proportion of estimation error size between DE and SE is about $\frac{1}{2} \approx \frac{0.86}{1.62}$. In all other cases, the proportion should be less than $\frac{1}{2}$. This suggests that $\frac{1}{3}$ should be an upper bound of $\beta(D, c)$ due to $\frac{1}{3}/\frac{2}{3} = \frac{1}{2}$, and therefore, $\beta(D, c) \in [0, \frac{1}{3}]$.

Based on the above analysis, we define c as a heuristic function with the following form:

$$c = \max\left(\frac{k \log_2 N}{\max(\Delta, 10^{-6})}, 2|\mu^V_{a^*}(D) - \mu^V_{a_L}(D)|\right), \quad (8)$$

where 10^{-6} is used to avoid the value of the denominator Δ becoming too close to 0, $2|\mu^V_{a^*}(D) - \mu^V_{a_L}(D)|$ is used to make $\beta(D, c) \in [0, \frac{1}{3}]$, and k is a constant independent of the characteristics of the problem that has to be determined empirically. In this paper, we set $k = 1$ because it leads to a much lower estimation error than DE on G2 and G3, as shown in Fig. 2. The average errors of WDE($k = 1$) are 0.10, 0.20, 0.16 on G1, G2, and G3, respectively, while the corresponding estimation errors are 0.01, 0.70, 0.35 in DE.

5.2 Roulette

Roulette is a bandit problem with 171 arms, consisting of 170 betting actions with an expected payout of \$0.947 per dollar and one walk-away action yielding \$0. Each betting action has an expected loss of \$$-0.053$ per play based on the assumption that a player bets \$1 every time. We recorded the mean action values over all betting actions as found by Q-learning (Q), bias-corrected Q-learning (BCQ), weighted Q-learning (WQ), double Q-learning (DQ), and weighted double Q-learning (WDQ) with different values of parameter c, and a heuristic value, denoted $c(k = 1)$ and defined by Eq. (8). Each trial involves a synchronous update of all actions. As the value of c increases, the degree of overestimation decreases. After $100,000$ trials, WDQ valued all betting actions \$12.87, \$2.93, \$$-0.11$, \$0.00 when $c = 1, 10, 100, c(k = 1)$, respectively. The estimates of the expected profit in all betting actions in WDQ($k = 1$), WDQ($c = 100$), BCQ, WQ, and DQ after $100,000$ trials are very close to the expected loss \$$-0.053$, all with absolute biases smaller than \$0.10.

5.3 Grid World

An $n \times n$ grid world problem has n^2 states (one state per cell in the grid) and four actions: {north, south, east, west}. Movement to adjacent squares is deterministic, but a collision with the edge of the world results in no movement. The initial state s_0 corresponds to the bottom left cell, and the goal state s_g corresponds to the top right cell. The agent receives a random reward of -30 or $+40$ for any action ending an episode from the goal state and a random reward of -6 or $+4$ for any action at a non-goal state. Since an episode starting from s_0 consists of $2(n-1)+1$ actions given an optimal policy, the optimal average reward per action is $\frac{5-2(n-1)}{2(n-1)+1} = \frac{7-2n}{2n-1}$. With a discount factor of $\gamma = 0.95$, $V^*(s_0) = 5\gamma^{2(n-1)} - \sum_{i=0}^{2n-3} \gamma^i$. Figure 3 shows the results of Q-learning and its variants on four $n \times n$ grid world problems.

In contrast with Hasselt's grid world problem, we assume stochastic reward at the goal state. Such a setting makes underestimating action values at the goal states negatively impact performance in DQ. In additon, the random reward at non-goal states makes the overestmates at non-goal states negatively impact performance in Q. As a result, both the Q and DQ agents often estimate that the action values at the goal state are lower than some action values at other states and do not move towards the goal state. Consequently, they performed poorly on all problems, as shown in Figure 3 (a–d). WDQ($c = 10$) and WDQ($k = 1$)[1] performed significantly better because their estimates of the maximum action values were much closer to the true values, e.g., at the initial state s_0 as shown in Figure 3 (e–h). On this domain, setting c to 10 is empirically a better choice in terms of the action-value estimate and policy quality than setting it adaptively by using Eq. (8), in part due to the shortcomings of approximating the constant in a bandit setting and deploying it in an MDP setting. We leave the design of a more effective heuristic to find an appropriate value of c in MDPs as a future research topic.

Figure 3 also shows the performance of WQ on the grid world domain. In terms of the average reward per action, WQ performs better than Q and DQ, but worse than WDQ($c = 10$) and WDQ($k = 1$). Compared with other methods, WDQ shows less max-operator bias.

5.4 Intruder Monitoring

An intruder monitoring task involves guarding danger zones using a camera so that if an intruder moves to a danger

[1]On this domain, $c = 10$ and $k = 1$ are not the best parameter values, and $c = 8$ and $k = 2$ can yield better empirical results.

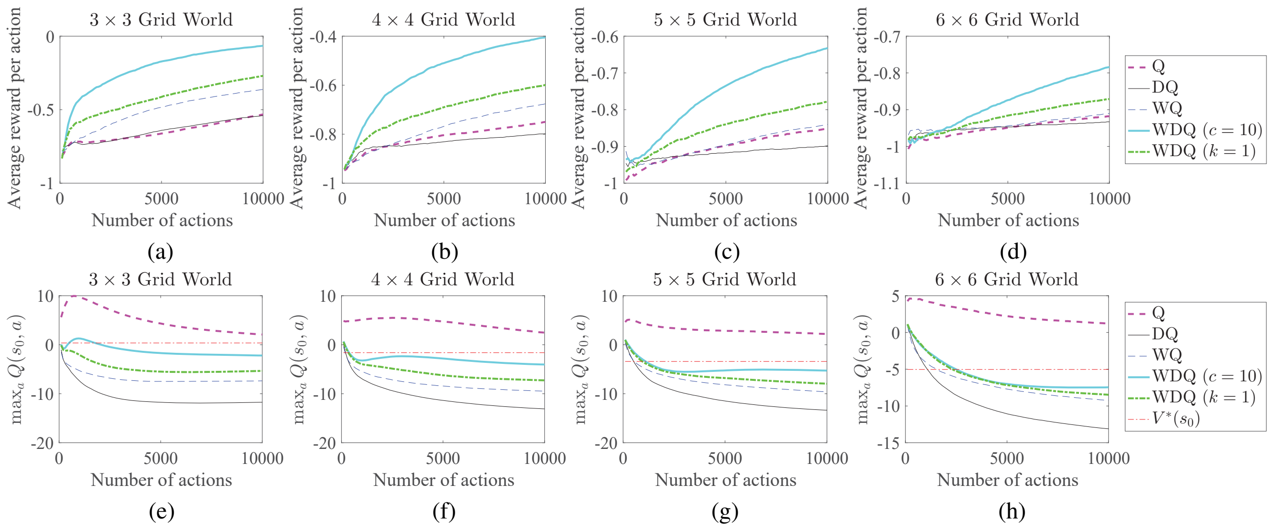


Figure 3: (a–d) The average reward per action and (e–h) the maximum action value in the initial state s_0 according to Q-learning and its variants on the $n \times n$ grid world problems, where n ranges from 3 to 6. Results are averaged over 1000 runs.

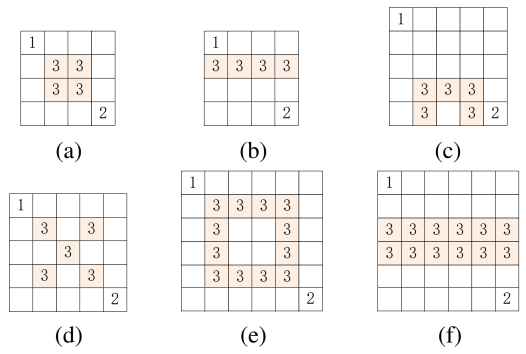


Figure 4: Maps of intruder monitoring problems: (a–b) two 4×4 maps with 256 states, (c–d) two 5×5 maps with 625 states, and (e–f) two 6×6 maps with 1296 states, where an intruder is initially located in 1, a camera initially in 2, and danger zones in 3.

zone, the camera points at that location. An $n \times n$ grid task has n^4 states. Each state corresponds to a unique position of the camera and intruder. At each time step, a camera or intruder has five actions to choose: {north, south, east, west, stay}. All movements are deterministic. The policy in the intruder is uniform random, but this is unknown to the camera. The system receives a reward of -10 for every visit of the intruder to a danger zone with no camera present, $+100$ when there is a camera present, and 0 otherwise. The discount factor γ is 0.95. Figure 4 shows the maps of six intruder monitoring problems. From Table 2 we can see that on these problems WDQ is robust and can perform better than both Q and DQ even when there is no randomness in the reward function. Compared with BCQ, WDQ can usually generate better policies due to its more accurate estimate of the maximum over action values. Since the only uncertainty is the behavior of the intruder, the small estimation errors in both Q and DQ sometimes have no significant negative impact in performance, making the improvement achieved by WDQ appear not as significant as on the grid world domain.

Table 2: The average reward per action over $200,000$ actions on the six intruder monitoring problems shown in Fig. 4. Results are averaged over 1000 runs.

Problems	Q	DQ	BCQ	WDQ(c)	WDQ(k)
Fig. 4 (a)	12.02	12.07	11.95	12.15	**12.30**
Fig. 4 (b)	14.35	14.48	14.35	14.69	**14.72**
Fig. 4 (c)	8.52	8.65	8.54	8.75	**8.79**
Fig. 4 (d)	8.32	8.31	8.33	**8.58**	8.53
Fig. 4 (e)	10.61	10.48	10.67	10.78	**10.79**
Fig. 4 (f)	9.64	9.54	9.75	9.82	**9.95**

WDQ(c): WDQ($c = 10$), WDQ(k): WDQ($k = 1$)

6 Conclusion

This paper presented a weighted double Q-learning algorithm to avoid the overestimation bias of action values inherent in regular Q-learning and the underestimation bias of action values inherent in double Q-learning. The parameter c is used to control the importance weight of the single and double estimates. We proposed a heuristic for selecting c based on insights from the empirical results in multi-arm bandit problems and used it to set the parameter adaptively. Empirically, we found that the new algorithm reduces the estimation error and performs well on a variety of MDP problems. In the future, it would be interesting to analyze the performance of weighted double Q-learning when using value function approximation on Atari video games [van Hasselt *et al.*, 2016; Mnih *et al.*, 2015].

Acknowledgments

This work was supported by National Natural Science Foundation of China (61502323), High School Natural Foundation of Jiangsu (16KJB520041), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[Bellman, 1957] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Bertsekas, 2007] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2007.

[Brafman and Tennenholtz, 2001] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 953–958, 2001.

[D'Eramo *et al.*, 2016] Carlo D'Eramo, Alessandro Nuara, and Marcello Restelli. Estimating the maximum expected value through Gaussian approximation. In *International Conference on Machine Learning (ICML)*, pages 1032–1040, 2016.

[Ernst *et al.*, 2005] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(1):503–556, 2005.

[Kearns and Singh, 1999] Michael J. Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 996–1002, 1999.

[Kochenderfer, 2015] Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[Lee and Powell, 2012] Donghun Lee and Warren B. Powell. An intelligent battery controller using bias-corrected Q-learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 316–322, 2012.

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

[Littman, 2015] Michael L. Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521:445–451, 2015.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Joel Veness Andrei A. Rusu, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[Strehl *et al.*, 2006] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 881–888, 2006.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[van Hasselt *et al.*, 2016] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 2094–2010, 2016.

[van Hasselt, 2010] Hado van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2613–2621, 2010.

[van Hasselt, 2011] Hado van Hasselt. *Insights in Reinforcement Learning*. PhD thesis, Center for Mathematics and Computer Science, Utrecht University, 2011.

[Watkins, 1989] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambridge, 1989.

[Wiering and van Otterlo, 2012] Marco Wiering and Martijn van Otterlo, editors. *Reinforcement Learning: State of the Art*. Springer, New York, 2012.