Received May 3, 2019, accepted June 5, 2019, date of publication June 13, 2019, date of current version July 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922706

Stochastic Double Deep Q-Network

PINGLI LV[1], XUESONG WANG[1,2], (Member, IEEE), YUHU CHENG[1,2], (Member, IEEE), AND ZIMING DUAN[3]

[1]School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China
[2]Xuzhou Key Laboratory of Artificial Intelligence and Big Data, Xuzhou 221116, China
[3]School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China

Corresponding author: Xuesong Wang (wangxuesongcumt@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772532.

ABSTRACT Estimation bias seriously affects the performance of reinforcement learning algorithms. The maximum operation may result in overestimation, while the double estimator operation often leads to underestimation. To eliminate the estimation bias, these two operations are combined together in our proposed algorithm named stochastic double deep Q-learning network (SDDQN), which is based on the idea of random selection. A tabular version of SDDQN is also given, named stochastic double Q-learning (SDQ). Both the SDDQN and SDQ are based on the double estimator framework. At each step, we choose to use either the maximum operation or the double estimator operation with a certain probability, which is determined by a random selection parameter. The theoretical analysis shows that there indeed exists a proper random selection parameter that makes SDDQN and SDQ unbiased. The experiments on Grid World and Atari 2600 games illustrate that our proposed algorithms can balance the estimation bias effectively and improve performance.

INDEX TERMS Estimation bias, deep reinforcement learning, maximum operation, double estimator operation, stochastic combination.

I. INTRODUCTION

Reinforcement learning (RL) is an important method for solving Markov decision process (MDP) problems. In RL, an agent learns the mapping from environment states to actions. Through interacting with environment, the agent obtains reward signals and modifies behavior selection policy. Therefore the agent can obtain the greatest reward from environment and maximize the long-term accumulative reward [1]. Many achievements have been made in the study of RL, and some classical algorithms, such as Q-learning [2], have been introduced. For the MDP problems with small state space and action space, the traditional RL algorithms are effective solutions. However, in most case the state space or action space is huge or continuous, which usually makes the traditional RL algorithms converge slowly or fail to converge to the optimal policy. Therefore, parameterizing the value function or policy and using function approximation have become the main solutions. Neural network is a popular used non-linear approximation function. Recently, Mnih *et al.* [3], [4] firstly proposed deep Q-network (DQN) by combining Q-learning and convolutional neural network [5], which has surpassed the level of human players on many Atari 2600 games. It opens up a new research field of deep reinforcement learning (DRL). In order to effectively represent the value function of reinforcement learning with deep neural network, two key approaches are applied in DQN. One is the experience replay mechanism, which is used to break the relevance among samples of RL. The other is the target network, which is used to select and evaluate actions and ensures the stability of computation of neural network. DQN realizes the deep computation of RL, which greatly expands the research scope and application scope of traditional RL, and makes DRL become the hot spot in the field of the artificial intelligence.

Following the DQN, many new DRL algorithms have emerged, such as double DQN (DDQN) [6], dueling DQN [7], deep recurrent Q-network [8], [9], deep deterministic policy gradient [10], prioritized experience replay [11], asynchronous advantage Actor-Critic [12], deep primal-dual policy learning [13], etc. In recent years, DRL has been successfully applied in many fields such as games [3], [4], [14]–[16], robot control [17]–[21], traffic signal control [22], power system [23], natural language processing [24], [25], and unmanned vehicles navigation [26], [27], etc.

The associate editor coordinating the review of this manuscript and approving it for publication was Chintan Amrit.

The maximum operation is a popular method used for estimating the value function, which often leads to overestimation of value function. This is also the reason of overestimation in DQN [6]. In order to eliminate overestimation, various methods have been proposed, such as the double estimator operation [28], the average estimator operation [29], [30] and the weighted estimator operation [31], [32]. van Hasselt *et al.* [6] proposed DDQN by using the double estimator operation. In DDQN, two different networks are used for action selection and action evaluation, respectively. DDQN can control overestimation effectively, but it may cause underestimation [6]. Anschel *et al.* [33] proposed averaged DQN (ADQN) based on the average estimator operation. In ADQN, the average of several previously learned Q values is calculated that is used as the target value function for selecting and evaluating actions. ADQN can reduce estimation bias, but needs more networks, occupies more storage resources and affects calculating efficiency. Lin *et al.* [34] combined episodic control with DQN, and proposed episodic memory deep Q-networks (EMDQN), which leverages episodic memory to supervise an agent during training. In EMDQN, it requires fewer interactions with the environment, generates better sample efficiency, and can also alleviate overestimation of DQN.

As we known, the maximum operation will lead to overestimation of value function, while the double estimator operation will lead to underestimation of value function. Therefore, can these two operations be combined to make up for their shortcomings? In view of this, we present a simple yet effective DRL algorithm called stochastic double deep Q-network (SDDQN), which is mainly based on double estimator framework and use the maximum operation and the double estimator operation together. At each step, we choose to use either the maximum operation or the double estimator operation with a certain probability, which is determined by a random selection parameter. The parameter gives the preference degree for each operation. Theoretical analysis shows that there indeed exists a proper selection parameter that makes SDDQN unbiased. SDDQN provides an effective means to solve the estimation bias problem, which can be easily extended to other RL algorithms that use the maximum operation or the double estimator operation. As a result, the stochastic double Q-learning (SDQ) is also considered in this paper, which is a tabular version of SDDQN. We evaluate our algorithms using the grid world and Atari 2600 games. Results show that both SDDQN and SDQ can effectively balance estimation bias and improve performance.

II. PRELIMINARIES

A. DEEP Q-NETWORK (DQN)

The action value function of RL is the expected discounted cumulative reward, which is defined by:

$$Q_t^{\pi}(s_t, a_t) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a; \pi] \tag{1}$$

where E(G) is the expectation of G. The optimal action value function is denoted as $Q^*(s, a) = \max_\pi Q^\pi(s, a)$, which satisfies the Bellman optimal equation [1]:

$$Q^*(s_t, a_t) = E[r + \gamma \max_a Q^*(s_{t+1}, a) | s_t, a_t] \tag{2}$$

The goal of value-based RL algorithms is to find an optimal policy $\pi^*(s) = \arg\max_{a \in A} Q^*(s, a)$ that maximizes the optimal action value function $\max_a Q^*(s, a)$.

Q-learning is one of the famous value-based RL algorithms [2], which estimates the optimal action value function through value iteration update. And the update rule is:

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t \\ + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \end{aligned} \tag{3}$$

where $\alpha_t(s_t, a_t)$ is the learning rate.

In tabular case, Q-learning uses a table to store all action values, which performs well on reinforcement learning problems with small-scale discrete state space and action space. However, many practical problems are large-scale or even continuous, and the tabular method cannot be effective in this case. Thus, the function approximation parametrized by θ is used to estimate the action value function. Generally, the parameter θ can be optimized by minimizing the following loss function,

$$L_t(\theta_t) = E(Y_t - Q(s_t, a_t; \theta_t))^2 \tag{4}$$

where $Y_t = r_t + \gamma \max_a Q(s_{t+1}, a; \theta_t)$ is the objective function. We can get the update rule of θ by gradient descent, which is

$$\theta_{t+1} = \theta_t + \alpha_t(Y_t - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{5}$$

By iteratively updating θ, the current action value function $Q(s_t, a_t; \theta_t)$ approximates the objective function Y_t.

As a deep learning version of Q-learning, DQN uses deep neural networks to approximate the action value function. In DQN, there are two different networks. One is the online network, used for estimating the current action value function $Q(s, a; \theta_t)$. Another is the target network, used for giving the objective function Y_t. Parameter θ_t of the online network is updated each step, while parameter θ_t^- of the target network is copied from the online network every N steps, and remains unchanged at other steps. The objective function used in DQN is:

$$Y_t^{DQN} = r_t + \gamma \max_a Q(s_{t+1}, a; \theta_t^-) \tag{6}$$

Furthermore, equation (6) can be transformed into:

$$Y_t^{DQN} = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t^-); \theta_t^-) \tag{7}$$

B. DOUBLE DEEP Q-NETWORK (DDQN)

The term $\max_a Q_t(s_{t+1}, a)$ in equation (3) means that the same action value function Q is used in the next state for both the maximum action selection and action evaluation, which may lead to overestimation on $Q(s, a)$. Therefore, Q-learning performs poorly under the stochastic environments. To avoid the overestimation of Q-learning, van Hasselt [28] proposed the double Q-learning (DQ) algorithm, in which two estimators $Q^0(s,a)$ and $Q^1(s,a)$ are used to separate the action selection and evaluation. At each step, one estimator is selected randomly to update, and the update formula is:

$$Q^i_{t+1}(s_t,a_t) = Q^i_t(s_t,a_t) + \alpha^i_t(s_t,a_t)(r_t + \gamma Q^{1-i}_t(s_{t+1},a^*) - Q^i_t(s_t,a_t)) \tag{8}$$

where $a^* = \arg\max_a Q^i_t(s_{t+1},a)$, $i \in \{0,1\}$.

Similarly, there is also the term $\max_a Q(s_{t+1},a;\theta^-_t)$ in DQN, which means that both action selection and evaluation use the same target network parameter θ^-_t, thus overestimation will also occur. Therefore, van Hasselt *et al.* [6] applied the idea of DQ into DQN, and proposed DDQN. In DDQN, the online network parameter θ_t is used for action selection, and the target network parameter θ^-_t is used for action evaluation. The objective function used in DDQN is:

$$Y^{DDQN}_t = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1},a;\theta_t);\theta^-_t) \tag{9}$$

The double estimator operation can reduce overestimation effectively. However, it may lead to underestimation on the action value function for using one network to evaluate the action that selected from the other network.

III. STOCHASTIC DOUBLE DEEP Q-NETWORK (SDDQN)

The maximum operation usually results in overestimating the action value function in DQN, while the double estimator operation may result in underestimation in DDQN. Therefore, instead of using only one single estimation operation, combining the maximum and double estimator operations together may make use of each operation's characteristic to make up for other's shortcomings so as to balance the estimation bias. In this paper, we combine these two operations by random selecting. In other words, we choose to use either the maximum operation or the double estimator operation with a certain probability at each step. Based on this idea, a new RL algorithm named stochastic double deep Q-network (SDDQN) is proposed, as shown in Algorithm 1.

In SDDQN, all steps are the same as DDQN or DQN, except the calculation of objective function Y_j. Suppose the random selection parameter λ_t $(0 \le \lambda_t \le 1)$ is given (we will discuss how to calculate this parameter later). When a random number ω is bigger than λ_t, we use the double estimator operation, i.e., we use the online network to select the maximum action $a^* = \arg\max_{a'} Q(s_{t+1},a';\theta_t)$ and the target network to evaluate this action. Otherwise, we use the maximum operation, i.e., we use the target network to select and evaluate the maximum action. Thus the objective function Y^{SDDQN}_j of SDDQN is defined as:

$$Y^{SDDQN}_j = \begin{cases} r_j + \gamma Q(s_{j+1}, \arg\max_{a'} Q(s_{j+1},a';\theta^-_t);\theta^-_t), \\ \qquad if\, \omega \le \lambda_t \\ r_j + \gamma Q(s_{j+1}, \arg\max_{a'} Q(s_{j+1},a';\theta_t);\theta^-_t), \\ \qquad otherwise \end{cases} \tag{10}$$

where θ_t and θ^-_t are the parameters of the online and target networks respectively.

Algorithm 1 SDDQN

```
1: Initialize replay memory $D$ to capacity $N$;
2: Initialize online network with random parameter $\theta$;
3: Initialize target network with parameter $\theta^- = \theta$.
4: **Repeat** (every episode)
5:     Initialize state $s_1$ and preprocessed $\phi_1 = \phi(s_1)$.
6:     **Repeat** (every step of each episode)
7:         Select $a_t$ from $\phi(s_t)$ based on $\varepsilon$-greedy policy;
8:         Execute action $a_t$, observe $r_t$, $s_{t+1}$;
9:         Store sample $(\phi_t, a_t, r_t, \phi_{t+1})$ into $D$;
10:        Sample random minibatch of samples $(\phi_j, a_j, r_j, \phi_{j+1})$
             from $D$;
11:        Set $Y_j = r_j$ if episode terminates at step $j+1$;
12:        Otherwise, calculate the random selection parameter
             $\lambda_t$; and
13:            calculate the objective function $Y_j$ according to
               equation (10);
14:        Perform a gradient descent step on $(Y_j - Q(\phi_j, a_j; \theta))^2$
           with respect to the network parameter $\theta$;
15:        Every $N$ steps reset target network parameter $\theta^- = \theta$.
16:    **Until**
17: **Until**
```

The random selection parameter λ_t gives the preference degree of selecting the double estimator operation or the maximum operation. In an extreme case, if $\lambda_t = 1$, which means the probability of selecting the maximum operation is 1, SDDQN will degenerate into DQN. Otherwise if $\lambda_t = 0$, SDDQN is equal to DDQN. Therefore, it can be easily seen that SDDQN makes full use of the advantages of these two operations, and combines DQN and DDQN into one algorithm.

Obviously, the random selection parameter λ_t will directly affect the performance of SDDQN. Proper λ_t will enable SDDQN to eliminate estimation bias. In fact, such λ_t does exist, which is shown in the following Theorem.

Theorem 1: Suppose $Q^*(s,a)$ be the optimal action value function, $Q(s,a;\theta)$ and $Q(s,a;\theta^-)$ be two unbiased estimators of $Q^*(s,a)$, i.e.,

$$E(Q(s,a;\theta)) = E(Q(s,a;\theta^-)) = E(Q^*(s,a)).$$

Let

$$\hat{Q}(s,a) = \begin{cases} \max_a Q(s,a;\theta^-) & if\, \omega \le \lambda \\ Q(s, \arg\max_a Q(s,a;\theta);\theta^-) & otherwise \end{cases} \tag{11}$$

where $\omega \in [0,1]$ be a random number. Thus there exists a parameter $\lambda \in [0,1]$ such that

$$E(\hat{Q}(s,a)) = \max_a E(Q^*(s,a)) \tag{12}$$

Proof: Let $a^* = \arg\max_a Q(s, a; \theta)$. When only the maximum operation $\max_a Q(s, a; \theta^-)$ is used, according to [28], we have

$$E(\max_a Q(s, a; \theta^-)) \geq \max_a E(Q(s, a; \theta^-)) = \max_a E(Q^*(s, a)).$$

And when only the double estimator operation is used, we have

$$E(Q(s, a^*; \theta^-)) \leq \max_a E(Q(s, a; \theta^-)) = \max_a E(Q^*(s, a)).$$

For convenience, we denote

$$E(\max_a Q(s, a; \theta^-)) - \max_a E(Q^*(s, a)) = \beta_1,$$
$$\max_a E(Q^*(s, a)) - E(Q(s, a^*; \theta^-)) = \beta_2.$$

If $\beta_1 = \beta_2 = 0$, then for any $\lambda \in [0, 1]$ we have

$$\begin{aligned} E(\hat{Q}(s, a)) &= E(\max_a Q(s, a; \theta^-))P(\omega \leq \lambda) \\ &\quad + E(Q(s, a^*; \theta^-))(1 - P(\omega \leq \lambda)) \\ &= \max_a E(Q^*(s, a))P(\omega \leq \lambda) \\ &\quad + \max_a E(Q^*(s, a))(1 - P(\omega \leq \lambda)) \\ &= \max_a E(Q^*(s, a)) \end{aligned}$$

Otherwise, we denote $\lambda = \beta_2/(\beta_1 + \beta_2)$, i.e., $P(\omega \leq \lambda) = \beta_2/(\beta_1 + \beta_2)$, and we have

$$\begin{aligned} E(\hat{Q}(s, a)) &= E(\max_a Q(s, a; \theta^-))P(\omega \leq \lambda) \\ &\quad + E(Q(s, a^*; \theta^-))(1 - P(\omega \leq \lambda)) \\ &= E(Q(s, a^*; \theta^-)) \\ &\quad + P(\omega \leq \lambda)[E(\max_a Q(s, a; \theta^-)) \\ &\quad - E(Q(s, a^*; \theta^-))] \\ &= \max_a E(Q^*(s, a)) - \beta_2 + \tfrac{\beta_2}{\beta_1+\beta_2}(\beta_1 + \beta_2) \\ &= \max_a E(Q^*(s, a)) \end{aligned}$$

Therefore, there exists $\lambda \in [0, 1]$ such that $\hat{Q}(s, a)$ is an unbiased estimator of $\max_a E(Q^*(s, a))$. □

The above analysis shows that proper values of λ_t that make SDDQN be unbiased exist in theory. But in practice, we cannot get the definite values of β_1, β_2 since the expected value is difficult to calculate accurately. Therefore, we will analyze how to estimate the approximate value of λ_t next. Because parameter θ^- keeps unchanged in each N steps, we can use $\hat{\beta}_{12} \triangleq \max_a Q(s, a; \theta^-) - Q(s, a^*; \theta^-)$ as the approximate estimate value of $\beta_1 + \beta_2 = E(\max_a Q(s, a; \theta^-)) - E(Q(s, a^*; \theta^-))$, which keeps constant in each corresponding N steps. Similarly, $\max_a E(Q^*(s, a))$ is an unknown constant, so $\beta_2 \approx \hat{\beta}_2 \triangleq \max_a E(Q^*(s, a)) - Q(s, a^*; \theta^-)$ also is a constant. Therefore, we can think λ_t as a constant in each N steps where θ^- keeps unchanged. Thus in the whole training process, λ_t can be thought as a piecewise staircase function. For RL problems that $\max_a E(Q^*(s, a))$ can be calculated accurately, we use $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$ as the approximate evaluate value of λ_t. Because $\max_a E(Q^*(s, a))$ cannot be obtained accurately, we adopt $\hat{\lambda}_t$ as the approximate evaluate value of λ_t. We assume that the initial value of $\hat{\lambda}_t$ is $\alpha(0 \leq \alpha \leq 1)$, and in τT steps its value changes from α to $\beta(0 \leq \beta \leq 1)$ piecewise linearly, and keeps be β from then on, where τ is the number of pieces and T is the steps of each piece. The calculating formula of $\hat{\lambda}_t$ is

$$\hat{\lambda}_t = \begin{cases} \alpha - (\alpha - \beta)\frac{k}{\tau} & (kT \leq t < (k+1)T \\ & \quad k = 0, 1, \cdots, \tau - 1.) \\ \beta & (t \geq \tau T) \end{cases} \tag{13}$$

If $\alpha = \beta$, $\hat{\lambda}_t$ is a constant of β, that is to say, we use all operations with fixed probability. If $\alpha > 0.5$, it indicates that SDDQN more tends to use the maximum operation at the beginning, otherwise, more tends to use the double estimator operation. If $\beta > 0.5$, it means that the degree of selecting the maximum operation is more than the double estimator operation after stabilization, otherwise the opposite. In extreme cases, if $\alpha = \beta = 1$, that is to say, the probability of selecting the maximum operation is 1, then SDDQN will degenerate into DQN; Otherwise, if $\alpha = \beta = 0$, SDDQN will degenerate into DDQN. We can set the corresponding values of α, β, τ and T according to the specific needs of each problem.

The essence of SDDQN is to present an idea to solve the estimation bias of action value function, which can be easily extended to other RL algorithms that use the maximum operation or the double estimator operation, such as Duel DQN, etc. As a result, a tabular version of SDDQN is given here, named stochastic double Q-learning (SDQ), which is shown in Algorithm 2.

Algorithm 2 SDQ

```
1: Initialize $Q^0(s, a)$, $Q^1(s, a)$
2: Repeat (every episode)
3:   Initialize $s_t$
4:   Repeat(every step of each episode)
5:     Select $a_t$ from $s_t$ base on $Q^0$ and $Q^1$ (e.g., $\varepsilon$-greedy in
       $Q^0 + Q^1$);
6:     Execute action $a_t$, observe $r_t$, $s_{t+1}$;
7:     Assign $i = 0$ or 1 randomly;
8:     Calculate the random selection parameter $\lambda_t$;
9:     Calculate $\hat{Q}^i_t(s_{t+1})$ according to equation (15);
10:    Update $Q^i_{t+1}(s_t, a_t)$ according to equation (14).
11:  Until
12: Until
```

In SDQ, there are two estimators, $Q^0(s, a)$ and $Q^1(s, a)$. At each step, we update one estimator randomly. And the updating formula of $Q^i(s, a)$, $i \in \{0, 1\}$ is,

$$Q^i_{t+1}(s_t, a_t) = Q^i_t(s_t, a_t) + \alpha^i_t(s_t, a_t)(r_t + \gamma \hat{Q}^i_t(s_{t+1}) - Q^i_t(s_t, a_t)) \tag{14}$$

where

$$\hat{Q}^i(s) = \begin{cases} \max_a Q^i(s, a) & \textit{if } \omega \leq \lambda_t \\ Q^i(s, \arg\max_a Q^{1-i}(s, a)) & \textit{otherwise} \end{cases} \tag{15}$$

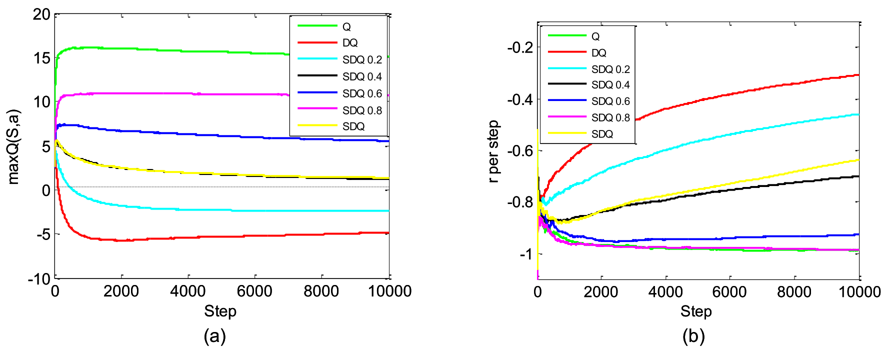


FIGURE 1. Results on the grid world by Q-learning, DQ, and SDQ with different $\hat{\lambda}_t$ as 0.2, 0.4, 0.6, 0.8 and $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$. (SDQ represents the result with $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$.) (a) The maximum action value of the initial state; (b) The average reward per step.

If the random number ω is less than or equal to λ_t, the maximum operation is selected, otherwise the double estimator operation is selected. If $\lambda_t = 1$, we always select the maximum operation, which means SDQ degenerates into Q-learning. Otherwise if $\lambda_t = 0$, we always select the double estimator operation. Here the double estimator operation in SDQ is different to the one in DQ. In DQ, another estimator is used to evaluate the maximum action selected by the current estimator, and the evaluation value is used to update the current estimator. However, in SDQ, we use the current estimator to evaluate the maximum action selected by another estimator, and the evaluation value is used to update the current estimator. Since $Q^0(s, a)$ and $Q^1(s, a)$ are two unbiased estimators of $Q^*(s, a)$, the performance of the double estimator operation in SDQ is the same as in DQ. The analysis of random selection parameter λ_t is same as above, and in practice we also use its estimate value $\hat{\lambda}_t$ for calculating.

As can be seen from the above analysis, comparing with DDQN (Double Q-learning), the amount of calculation increased in SDDQN (SDQ, respectively) is very small,which is caused by the calculating of $\hat{\lambda}_t$. And the sample data used in SDDQN (SDQ) is the same as that in DDQN (Double Q-learning, respectively), so SDDQN (SDQ) is as data valid as DDQN (Double Q-learning, respectively).

IV. EXPERIMENTS

In this section, we first demonstrate the experiments of SDQ on the Grid World, of which the optimal action value function can be calculated exactly, and compare with Q-learning and DQ. Additionally, we present experiments of SDDQN on four Atari 2600 games with OpenAI Gym platform, and compare with DQN and DDQN.

A. GRID WORLD

van Hasselt [28] used a 3 × 3 grid world to show the over-estimation of Q-learning and underestimation of DQ. For comparison, we also use the same grid world to test the performance of SDQ. In the 3 × 3 grid world, the initial state is in the lower left cell, and the goal state is in the upper right cell. Each state has four optional actions: up, down, left and right. If an action selected by the agent causes it to leave the grid world, the agent remains in the original state after taking the action. Before reaching the goal, the agent receives a random reward of -12 or $+10$. In the goal state, the agent receives reward of $+5$ for every action and ends an episode. Discount factor γ is 0.95 and learning rate $\alpha_t(s, a)$ is $1/n_t(s, a)$, where $n_t(s, a)$ represents the number of updates of the action value function on each state-action pair (s, a). For SDQ and DQ, if update Q^0 then $n_t(s, a) = n_t^0(s, a)$, otherwise $n_t(s, a) = n_t^1(s, a)$. The exploration policy is ε-greedy with $\varepsilon(s) = 1/\sqrt{n(s)}$, where $n(s)$ is the number that state s has been visited. The algorithm is training 10,000 steps and the results are averaged over 1,000 experiments. According equation (1), we can calculate the theoretical optimal action value for each state exactly. For example, the optimal policy of initial state includes five steps, therefore the optimal average reward per step is $+0.2$, and the theoretical optimal action value of the initial state is $5\gamma^4 - \sum_{k=0}^{3} \gamma^k \approx 0.36$ (See the horizontal dashed line in Fig. 1 and Fig. 2)

Here, we mainly analyze the influence of λ_t on the performance of SDQ. Since $\hat{\beta}_2$, $\hat{\beta}_{12}$ can be calculated accurately, we firstly use $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$ as the approximate estimate value of λ_t, and the experimental results of the optimal action value max $Q(s, a)$ and average reward per step of initial state are shown in Fig. 1, which is labeled as SDQ. Fig. 1 also shows the results of $\hat{\lambda}_t$ being the fixed value of 0.2, 0.4, 0.6 and 0.8. And the results of Q-learning and DQ are also given. It can be observed from Fig. 1 (a) that, the optimal action value obtained by using $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$ is extremely close to the theoretical value, which means that the approximate estimate formula $\hat{\lambda}_t = \hat{\beta}_2/\hat{\beta}_{12}$ is reasonable and SDQ is effective to eliminate the estimation bias. Meanwhile, the result of $\hat{\lambda}_t$ being 0.4 is almost the same as the formula, which shows that SDQ can also obtain satisfactory result with the proper fixed value of $\hat{\lambda}_t$. Additionally, the larger the value of $\hat{\lambda}_t$ is, the closer the effect of SDQ is to the one of Q-leaning, otherwise the closer to the one of DQ. SDQ mainly focuses on how to balance the estimation bias, thus the effect on the average reward per step may be not as well as DQ, but which

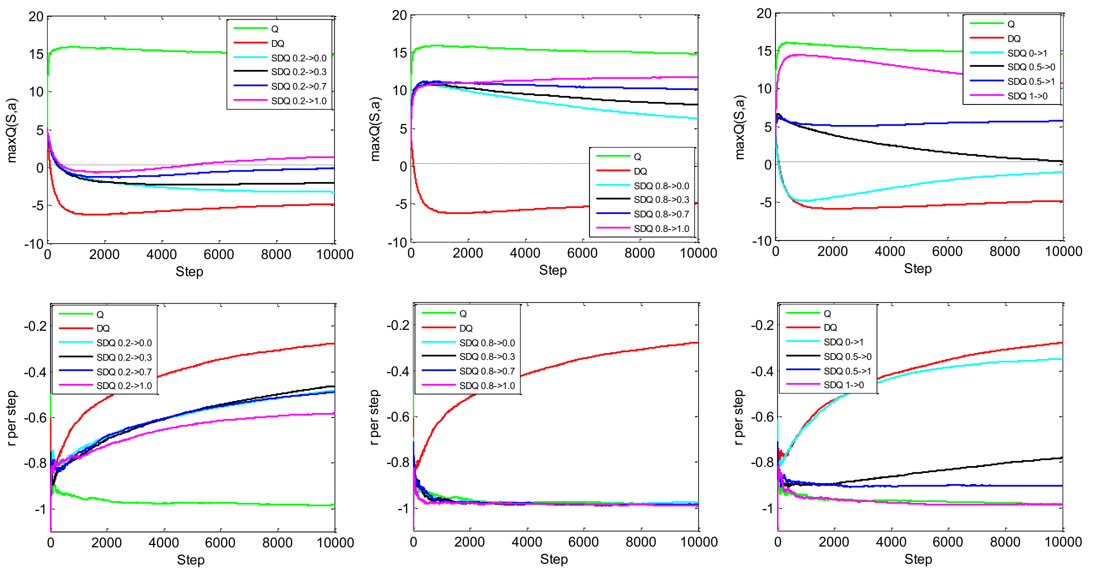


FIGURE 2. Results on the grid world by Q-learning, DQ, and SDQ with $\hat{\lambda}_t$ calculated by equation (13). The top row shows the maximum action values of the initial state, and the bottom row shows the average rewards per step. (SDQ 0.2->0.7 means that $\alpha = 0.2$ and $\beta = 0.7$).

is still better than Q-learning. And the smaller the fixed value of $\hat{\lambda}_t$, the better the algorithm performance.

Furthermore, we have also tested SDQ for $\hat{\lambda}_t$ being calculated according to equation (13). Here, let $\tau = 25$ and $T = 200$, that is to say, $\hat{\lambda}_t$ changes piecewise linearly at the first 5000 steps, and then kept constant. The control parameters are set as the follows: α is 0.2 and 0.8, β is 0, 0.3, 0.7 and 1. Additionally, we also show the experimental results when α is 0, 0.5 and 1, and β is 0 and 1. Fig. 2 shows the experimental results including the optimal action value of the initial state and the average reward per step. It can be observed that: 1) When α is 0.2, the optimal action values of SDQ for different values of β are better than those of DQ and Q-learning. Especially when β is larger, the effect is more obvious and can reach the theoretical value. While balancing the estimation bias effectively, good results of SDQ in the average reward per step have also been achieved; 2) When α is 0.8, the optimal action values of SDQ are better than that of Q-learning, and the smaller the value of β, the better the result is. However, it is always overestimated within 10,000 steps, which results in no improvement in the average reward per step. The reason is mainly that the maximum operation is used more frequently in the initial steps, and a more serious overestimation has been accumulated, which has no obvious improvement even if it have become completely the double estimator operation after 5000 steps; 3) In the extreme case such as α is 0 and β is 1, the improvement of SDQ on the optimal action value is very obvious, and the final result is close to the theoretical value. At the same time, the average reward per step is almost as good as DQ; 4) When α is 0.5, the optimal action values are not overestimated any more than ever, even if it has been the maximum operation after 5000 steps, which means that the double estimator operation selected in the previous steps has suppressed the overestimation phenomenon very well. And the result when β equals 0 is particularly well, which achieves the theoretical optimal value. However, in the average reward per step, although it is better than that of Q-learning, the improvement is not obvious.

All above results illustrate that the value of λ_t can directly affect the performance of SDQ. And whether it is a formula or a fixed value, we can always find some proper values of λ_t, so that SDQ can effectively eliminate the estimation bias and improve the performance.

B. ATARI 2600 GAMES

We select four Atari 2600 games on the OpenAI Gym platform to test the proposed SDDQN, which are Asterix, Zaxxon, Breakout and Seaquest. Asterix is an adventure game. It needs both risky and strategic. Continuous overestimation will result in lower scores and lower learning ability of the algorithm during later stage of the game. Therefore, Asterix is selected to test whether there is overestimation bias in the algorithm. Zaxxon is a shooting game, and underestimation will seriously affect the performance of the algorithm. It is very suitable for testing whether the algorithm has underestimation bias. For the game of Breakout, some benchmark algorithms such as DQN can easily get stable policy on it and it is a frequently-used game for test. Seaquest is a type of complex games with rare rewards and partially observable environment, and many algorithms are learning unstable on it, so we use it to test the stability of algorithm.

The network structure used in this paper is the same as reported in previous study [4], and the input images are preprocessing firstly. The specific parameters are as below:

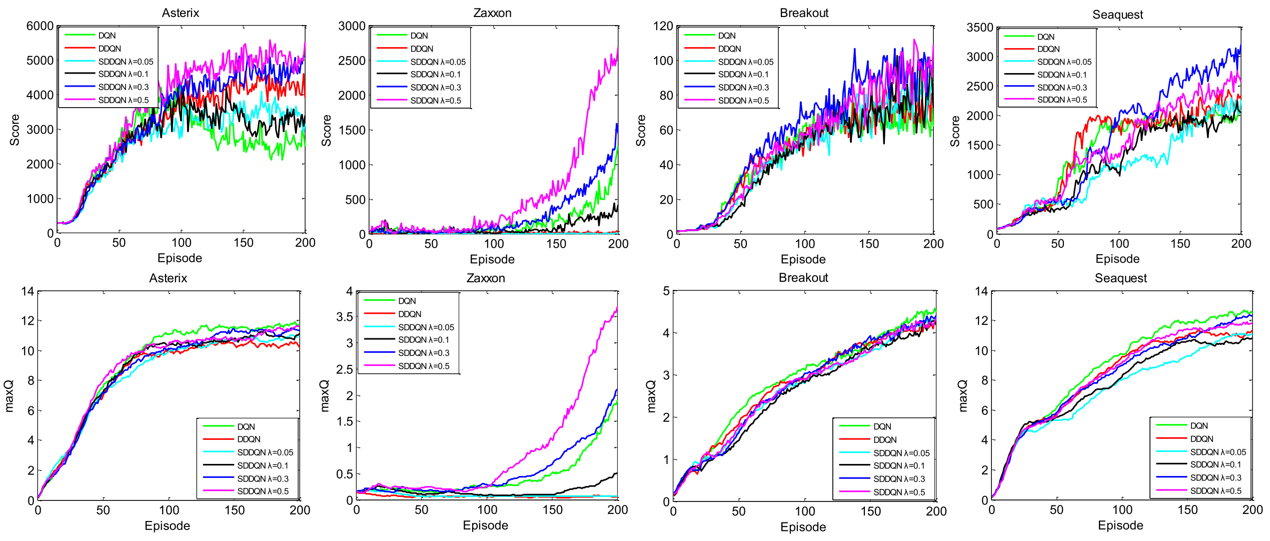


FIGURE 3. DQN, DDQN, and SDDQN performance (top row), and maximum action value estimates (bottom row) in four Atari games, with different $\hat{\lambda}_t$ as 0.05, 0.1, 0.3, 0.5.

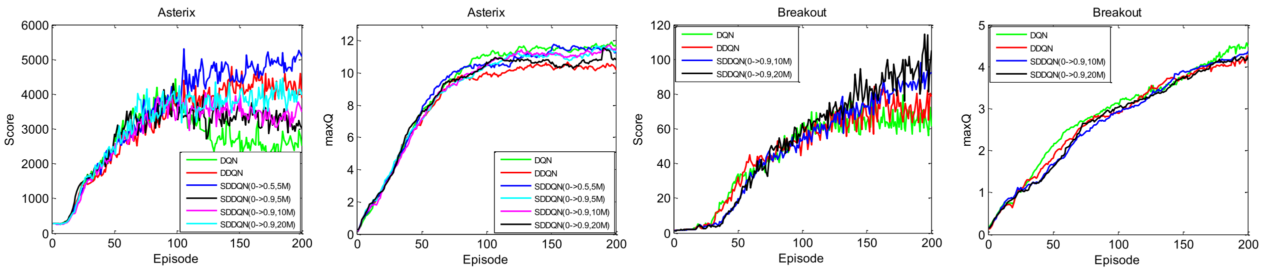


FIGURE 4. DQN, DDQN, and SDDQN performance and maximum action value estimates in two Atari games, with $\hat{\lambda}_t$ calculated by equation (13).

the discount factor is $\gamma = 0.99$, learning rate $\alpha = 2.5 \times 10^{-4}$, the exploration rate of ε-greedy policy is $\varepsilon = 0.05$. The number of training episodes is 200, training steps of each episode is 200,000, learning interval is 1. Copy the parameters of the online network to the target network after every 40,000 steps. The size of mini-batch is 32, and size of the experience replay memory is 1,000,000. The RMSProp method is used to update the network parameters with momentum parameter 0.95. We firstly run 10,000 steps to store random samples to the experience replay memory.

Firstly, we select four fixed values of $\hat{\lambda}_t$ for the experiment, which are 0.05, 0.1, 0.3 and 0.5. The experimental results are shown in Fig.3, where the top row is the scores of SDDQN, DDQN and DQN on four games, and the bottom row is the maximum action value estimates. We can conclude that, when $\hat{\lambda}_t$ is equal to 0.3 and 0.5, the scores of SDDQN on four games are better than those of DQN and DDQN, especially on Zaxxon game. As a shooting game, Zaxxon needs more optimistic policy to get higher scores. In this game, DDQN has not learned any useful policy due to underestimation. While the scores of SDDQN are not good either when $\hat{\lambda}_t$ is small, proper values of $\hat{\lambda}_t$ make SDDQN not only unaffected by underestimation, but also outperformed DQN algorithm obviously in score. The maximum action value curves of Zaxxon also appropriately reflect the characteristics of this game, which mean that DQN is not overestimated on this game, but not enough. On the game of Asterix, after about 100 Episodes, the score of DQN begins to decrease as a result of overestimation, but SDDQN is not affected by overestimation, and when $\hat{\lambda}_t$ equals 0.3 and 0.5, the scores of SDDQN are significantly higher than that of DDQN. On Seaquest game, with a wave of rapid growth after about 80 episodes, the scores of DQN and DDQN begin to stabilize. But the trend of scores of SDDQN in the whole learning episodes have been upward, indicating that the score ability of SDDQN is continuously improved, and the learning is more stable. Especially when $\hat{\lambda}_t$ equals 0.3 and 0.5, after about 100 episodes, the scores of SDDQN have exceeded those of DQN and DDQN, and the upward trends are obvious. When $\hat{\lambda}_t$ is 0.05 and 0.1, although SDDQN performs poorly in the early episodes, their scores also reach the levels of DQN and DDQN within about 200 episodes. On the Asterix, Breakout and Seaquest games, the difference of maximum action value curves among DQN, DDQN and SDDQN are not obvious, but proper values of $\hat{\lambda}_t$ make SDDQN obtain better scoring ability.

Additionally, we also test SDDQN with $\hat{\lambda}_t$ calculated according to equation (13) on two games, and the results are shown in Fig. 4. Here we set $T = 1$, i.e., $\hat{\lambda}_t$ changes at each step. We experiment on two games in the case of α be 0 and β be 0.9 when $\tau = 10M$ and $\tau = 20M$. On Asterix game, we also give the experimental results when α is 0, β is 0.5 and 0.9, and $\tau = 5M$. For the score of SDDQN on Asterix, it is the best case when β is 0.5 and $\tau = 5M$. When α is 0 and β is 0.9, the smaller the value of τ, the worse the score is, indicating that the effect of overestimation is greater. For the score of Breakout, SDDQN performs better than the other two. The results also show that the same change tendency of $\hat{\lambda}_t$ has different effects on different games

The above experimental results show that, for RL problems which are obviously suffering from the estimation bias, the stochastic combination of the maximum and double estimator operations can make full use of the advantages of each operation, and make the estimate of the maximum action value more reasonable, so as to achieve the performance effect of exceeding any single operation. However, choosing proper value of λ_t is the key to improving the performance of SDDQN.

V. CONCLUSION

In this paper, a new algorithm SDDQN is proposed to balance the estimation bias and improve the performance of the algorithm, which takes full account of advantages of the maximum operation and the double estimator operation and combines them with the idea of random selection. The tabular algorithm SDQ is also proposed which is designed with the same idea. Our algorithms provide a simple and effective way to solve the estimation bias problem, which can be easily extended to other RL algorithms using the maximum operation or the double estimator operation. The probability of choosing each operation is defined by the random selection parameter, which directly affects the performance of the algorithm. The random selection parameter λ is analyzed in the paper. It is shown that there are proper values of λ to make the new algorithm unbiased. Because the theoretical value of the random selection parameter is not easy to obtain, the estimate formula is given in this paper. The experimental results on the grid world and Atari 2600 games show that new algorithms can effectively balance the estimation bias and achieve good performance by choosing proper values of the random selection parameter. Because of the importance of the random selection parameter, although we have done some analysis on this parameter here, the parameter adjusted accurately within the learning process will improve the performance of the algorithm to a greater extent. Therefore, how to select more accurate random selection parameter is a key issue worthy of further study, such as adaptive selection can be considered.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[2] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," presented at the NIPS Deep Learn. Workshop, Lake Tahoe, NV, USA, 2013. [Online]. Available: https://arxiv.org/pdf/ 1312.5602v1.pdf

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[5] Y. Bengio, "Deep learning of representations: Looking forward," in *Proc. SLSP*, Tarragona, Spain, 2013, pp. 1–37.

[6] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, Phoenix, AZ, USA, 2016, pp. 2010–2094.

[7] Z. Wang, N. D. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, New York City, NY, USA, vol. 4, 2016, pp. 2939–2947.

[8] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable MDPs," in *Proc. AAAI-SDMIA*, Arlington, VA, USA, Nov. 2015, pp. 29–37.

[9] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent Q-networks," 2016, *arXiv:1602.02672*. [Online]. Available: https:// arxiv.org/pdf/1602.02672.pdf

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," presented at the ICLR, San Juan, Puerto Rico, 2016. [Online]. Available: https://arxiv.org/pdf/1509.02971.pdf

[11] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," Presented at ICLR, San Juan, Puerto Rico, 2016. [Online]. Available: https://arxiv.org/pdf/1511.05952.pdf

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, New York City, NY, USA, vol. 4, 2016, pp. 2850–2869.

[13] W. S. Cho and M. Wang, "Deep primal-dual reinforcement learning: Accelerating actor-critic using bellman duality," 2017, [Online]. *ariv:1712.02467*. [Online]. Available: https://arxiv.org/abs/1712.02467

[14] M. Moravčik, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, May 2017.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp: 484–489, Jan. 2016.

[17] J. Li, B. Kiumarsi, T. Chai, F. L. Lewis, and J. Fan, "Off-policy reinforcement learning: Optimal operational control for two-time-scale industrial processes," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4547–4558, Dec. 2017.

[18] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," presented at the ACRA, Canberra, Australia, Dec. 2015. [Online]. Available: http://www.araa.asn.au/acra/acra2015/ papers/pap168.pdf

[19] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. ICCV*, Santiago, Chile, Dec. 2015, pp. 2488–2496.

[20] J. Škach, B. Kiumarsi, F. L. Lewis, and O. Straka, "Actor-critic off-policy learning for optimal control of multiple-model discrete-time systems," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 29–40, Jan. 2018.

[21] B. Luo, Y. Yang, and D. Liu, "Adaptive Q-learning for data-based optimal output regulation with experience replay," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3337–3348, Dec. 2018.

[22] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," 2018, *arxiv:1803.11115*. [Online]. Available: https://arxiv.org/abs/1803.11115

[23] Z. Wang, Y. Chen, F. Liu, Y. Xia, and X. Zhang, "Power system security under false data injection attacks with exploitation and exploration based on reinforcement learning," *IEEE Access*, vol. 6, pp. 48785–48796, 2018.

[24] H. Guo, "Generating text with deep reinforcement learning," pretended at the NIPS, Montreal, QC, Canada, Dec. 2015. [Online]. Available: https://arxiv.org/pdf/1510.09202v1.pdf

[25] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, Vancouver, BC, Canada, May 2013, pp. 6645–6649.

[26] S.-M. Hung and S. N. Givigi, "A Q-Learning approach to flocking with UAVs in a stochastic environment," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 186–197 Jan. 2017.

[27] M. Wei, S. Wang, J. Zheng, and D. Chen, "UGV navigation optimization aided by reinforcement learning-based path tracking," *IEEE Access*, vol. 6, pp. 57814–57825, 2018.

[28] H. V. Hasselt, "Double Q-learning," in *Proc. NIPS*, Vancouver, BC, Canada, 2010, pp. 2613–2621.

[29] L. Kocsis and C. Szepesvári, "Bandit based Monte–Carlo planning," in *Machine Learning: ECML*. Berlin, Germany: Springer, 2006, pp. 282–293.

[30] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[31] C. D'Eramo, M. Restelli, and A. Nuara, "Estimating maximum expected value through Gaussian approximation," in *Proc. ICML*, New York, NY, USA, vol. 48, Jun. 2016, pp. 1032–1040.

[32] T. Imagaw and T. Kaneko, "Estimating the maximum expected value through upper confidence bound of likelihood," in *Proc. TAAI*, Taipei, Taiwan, Dec. 2017, pp. 202–207.

[33] O. Anschel, N. Baram, and N. Shimkin, "Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning," in *Proc. ICML*, vol. 1, Sydney, NSW, Australia, 2017, pp. 240–253.

[34] Z. Lin, T. Zhao, G. Yang, and L. Zhang, "Episodic memory deep Q-learning," in *Proc. IJCAI*, Stockholm, Sweden, 2018, pp. 2433–2439.

PINGLI LV received the master's degree from the China University of Mining and Technology, Xuzhou, China, in 2011.

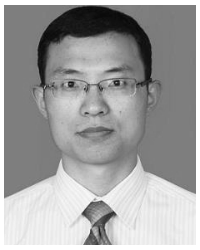
Her main research interests include reinforcement learning and deep reinforcement learning.

XUESONG WANG (M'15) received the Ph.D. degree from the China University of Mining and Technology, Xuzhou, China, in 2002, where she is currently a Professor with the School of Information and Control Engineering.

Her main research interests include machine learning, bioinformatics, and artificial intelligence.

YUHU CHENG (M'15) received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2005.

He is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology. His main research interests include machine learning and intelligent systems.



ZIMING DUAN received the Ph.D. degree from the China University of Mining and Technology, Xuzhou, China, in 2011, where he is currently an Associate Professor with the School of Mathematics.

His main research interests include graph model and reinforcement learning.