**International Journal of Robotics and Automation (IJRA)**
Vol. 7, No. 1, March 2018, pp. 39~47
ISSN: 2089-4856, DOI: 10.11591/ijra.v7i1.pp39-47


# An Actor-critic Algorithm Using Cross Evaluation of Value Functions

**Hui Wang, Peng Zhang, Quan Liu**
School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, China

**Article Info**

***Article history:***

Received Sep 4, 2017
Revised Dec 5, 2017
Accepted Jan 11, 2018

***Keyword:***

Actor-critic
Continuous spaces
Cross evaluation
Reinforcement learning

**ABSTRACT**

In order to overcome the difficulty of learning a global optimal policy caused by maximization bias in a continuous space, an actor-critic algorithm for cross evaluation of double value function is proposed. Two independent value functions make the critique closer to the real value function. And the actor is guided by a crossover function to choose its optimal actions. Cross evaluation of value functions avoids the policy jitter phenomenon behaved by greedy optimization methods in continuous spaces. The algorithm is more robust than CACLA learning algorithm, and the experimental results show that our algorithm is smoother and the stability of policy is improved obviously under the condition that the computation remains almost unchanged.



***Corresponding Author:***
Quang Liu,
School of Computer Science and Technology,
Soochow University,
No. 1 Shizi Street, Suzhou, Jiangsu 215000, China
Email: quanliu@suda.edu.cn

## 1. INTRODUCTION

Temporal difference algorithm in reinforcement learning (RL) usually uses a maximizing operation to solve the optimal policy. Whether the off-policy Q-learning or in-policy SARSA, the maximization is required to find the optimal action of the current state, and based on this action, algorithmes continuously update the state-action value functions. The value functions calculated always have some deviations, which are usually referred to as the maximization bias. In some special circumstances, these deviations may seriously affect the learning efficiency of the agent.

The maximization bias may shift the learning goals, making the policy calculation fall into a local optimal. The policy learned at this time is optimal in the adjacent policy space, but it is not the optimal in the entire policy space. The local optimal policy occurs because the agent does not adequately access the state and action space, and can not accumulate enough sample data related to the optimal policy. In a learning environment of large-scale continuous state space or action space, reinforcement learning algorithm needs to avoid the improper optimization guidance which may present an incorrect direction about the optimal policy. In such situation, the maximization bias shows an obvious side effect. At the same time, the learning algorithm in a continuous space is more likely to produce curse of dimensionality, which makes the exloration time in both policy and state space rise along the exponential curve.

The optimization of the function approximator is a common way to break the curse of dimensionality in continuous space. A linear approximation was first used by Samuel to implement an artificial checker player [1]. Sutton combined the tempral difference learning method enhanced by eligible trajectory with a linear function approximator, and uses the gradient descent method to solve the approximate value functions [2]. Engel used Gaussian process to model the value function, and proposed a temporal difference reinforcement learning method with Gaussian process [3]. Their work demonstrates that optimized function approximators can achieve excellent experimental results in continuous state and action spaces.

The actor-critic approach [4] combines the advantages of the value function method and the policy search method, while storing both value functions and policies. When the agent selects an action, it only needs to select it based on the stored policy without knowing of the value function. When an immediate reward is obtained from the environment, the agent updates the value function and maintains the currently stored policy based on the change of the value function.

In the continuous state and action space, the actor critic method can avoid the convergence to a local optimal by using only the value function, and also solve the larger estimation bias problem in most policy search methods [5,6]. In recent years, the use of actor critic methods to solve reinforment learning problems in continuous space has become a research hotspot.

Sutton proposed a method of applying function approximation to policy gradients, which depends on a dedicated action representation[7]. The method first introduces a policy value function, and defines the objective of continuous space reinforcement learning as the maximization of policy value function. Peters applied the natural gradient method to function approximation. He combined the temporal difference with least squares algorithm in reinforcement learning, and designed natural actor-critic (NAC) algorithm [8,9]. Hasselt used the action difference to improve policy parameters to evaluate the pros and cons of actions, and proposed a continuous actor-critic learning automaton (CACLA) [10].

Wierstra applied the natural gradient method and the evolutionary policy method to the policy update, and proposed the natural evolutionary strategies (NES) [11,12]. Busoniu used cross entropy to optimize the parameters of the basis functions, and proposed a cross-entropy optimization method [13,14]. Martin used the k-nearest neighbor classification to discretize the space, and proposed a temporal difference algorithm based on k-nearest neighbor classification [15]. Lillicrap used a policy gradient to study the deep reinforcement learning problem and proposed a deep deterministic policy gradient algorithm [16]. Gu used a model learning method to improve the convergence rate in continuous space, and proposed a continuous space deep Q-learning algorithm based on model acceleration [17]. Khamassi applied the meta-learning method to the exploration of the parameteres in continuous spaces, and proposed an actor-critic learning method based on meta-learning [18].

## 2. ACTOR-CRITIC METHOD

According to the selection of actions, reinforcement learning method can be divided into three categories: actor-only, critic-only and actor-critic. The actor-only method does not estimate the value function. Agent follows its current policy to interact the environment. The immediate reward aquaired from the enviroment is used directly to optimize current policy.

The critic-only approach does not need to maintain a policy. The policy is calculated from the current value function by interacting with the environment. The current value function is continually optimized using the reward obtained. Unlike actor-only and the critic-only approaches, the actor-critic approach consists of both the actor and the critic, which need to maintain value functions and policy at the same time. The actor used to select actions, and the critic can comment on the selected action good or bad. The way that actors choose actions is not based on the current value function, but their own policy. Critics' comments generally take the form of temproal difference errors, which is calculated from the current value function. The temporal difference error is the only output of the critic and drives all the learning between the actor and the critic [19, 20]. The actor's algorithm structure is shown in Figure 1.

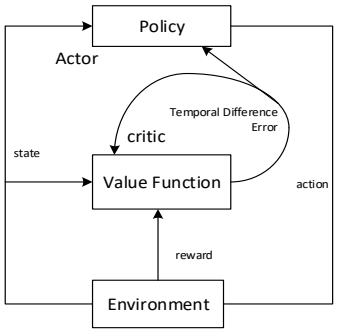



Figure 1. The framework of actor-critic method

The traditional actor-critic approach is used in discrete state and action spaces. The actor selects a performing action in the current state based on a lookup table. The lookup table stores the corresponding preference $p(x,u)$ for each state action pair. $x,u$ represent state and action. According preference $p$, the selection probability of each action can be computed under the current state. The action selection method is Gibbs softmax. As shown in Equation 1.

$$h(x,u)=\frac{e^{p(x,u)}}{\sum_{u'\in U} e^{p(x,u')}} \tag{1}$$

Critic uses temporal difference error to comment on the quality of actions. Once the critic gets the action, the temporal difference error is calculated from the value function. The temporal difference error $\delta$ is calculated with equation (2). $V$ is state value function, $\gamma \in [0,1)$ is a discount and $r$ represents an immediate reword.

$$\delta_t = r_{t+1} + \gamma V(x_{t+1}) - V(x_t) \tag{2}$$

Preference $p$ can be updated iteratively with equation (3). $\beta$ is a step factor.

$$p(x_t,u_t) \leftarrow p(x_t,u_t) + \beta\delta_t \tag{3}$$

## 3. CROSS EVALUATION OF VALUE FUNCTIONS

In temporal difference algorithm, the greedy method, which depends on the maximizing operation, is used for selecting actions. Because of the maximization bias, a part of value function will be over-evaluated and be easily affected by the lack of exploration of the environment. This makes the learning policy often trap into a local optimal. In particular, for the off-policy Q-learning algorithm, its evaluation policy never explores, so it is easier to fall into a local optimal policy [21]. Exploration can reduce the maximization bias, but the overall efficiency is low. The adoption of multiple cross evaluated value functions for temporal difference learning can decrease the maximization bias [22].

In short, cross-evaluation value function usually uses two sets of value functions for temporal difference learning. Unlike the usual temporal difference algorithm, there are two performers in the learning process, with two sets of independent evaluation functions. When using the action state value function, the evaluation functions are $Q_1(u)$ and $Q_2(u)$, action $u \in U$ (set of actions). For each evaluation function $Q_1(u)$, the optimal action is calculated with equation (4).

$$u^* = \max_u Q_1(u) \tag{4}$$

In another evaluation function $Q_2(u)$, it is needed to check the optimal action obtained by $Q_1(u)$. The evaluation follows equation (5).

$$Q_2(u^*) = Q_2\left(\max_u Q_1(u)\right) \tag{5}$$

When condition $E\left[Q_2(u^*)\right] = q(u^*)$ is satisfied, the estimation is unbiased. $q(u)$ represents the true value function. The same method is used to cross examination the first set of action state value function. The calculation follows equation (6).

$$Q_1(u^*) = Q_1\left(\max_u Q_2(u)\right) \tag{6}$$

Agent select actions based on two intertwined evaluation functions $Q_1$ and $Q_2$. Most occasions $Q_1(u)+Q_2(u)$ is used to replace current value functions. Temporal difference learning method like this is the cross evaluation function approach. Using this method for evaluation of the optimal action, only one of the evaluation functions is updated at each time step, and the action evaluated by another evaluation function is an important learning parameter. This method added one more evaluation function and requires more storage. But the computational complexity remains almost same as the non-cross evaluation approach. The new learning algorithm still satisfies the greedy principle, and the selection of actions is related to the optimal action in the evaluation policy.

The update of value function is constrained by another value function. If one of the value functions evaluates the error due to the maximization bias, the next state-action pair in another set of evaluations will correct the deviation. In this way, another set of evaluation values will also push the current value function out of the local optimal state, reducing the maximization bias while continuingly selecting the optimal action. This method is particularly effective in environments where local optimal policies are prone to occur.

## 4. AN ACTOR-CRITIC ALGORITHM BASED ON CROSS EVALUATION OF VALUE FUNCTIONS (DVCAC)

The traditional cross evaluation of value function algorithm constructs two sets of evaluation functions. By adjusting the update equation, these two evaluation functions can restrain each other, so that the policy computed is balanced. This method can reduce the maximization bias and make the policy get out of the local optimal faster, and speed up the convergence of the algorithm [2]. The cross evaluation of value function algorithm can be used in action-critic algorithm in continuous space, for reducing the possibility of local optimal situations.

In solving reinforcement learning problems, the state value function or the state action value function is used to evaluate the policies. Using the state value function can reduce the amount of storage and computation required for parameter updates, but the calculation of the current optimal action is significantly increased. In the actor-critic algorithm, the optimal action is obtained directly through the current cached policy, and no more value function is needed to compute it, so in actor-critic algorithm in a continuous space, current policy is evaluated by using the state value function with less storage capacity.

In continuous space, state-valued functions are approximated by linear functions. The representation method is shown in equation(7).

$$V(x)=\theta^{\mathrm{T}}\cdot\phi(x) \tag{7}$$

And the updating of the policy parameter is shown in equation(8).

$$\theta=\theta+\alpha\delta\cdot\phi(x) \tag{8}$$

The value function relationship between the current state and the next state is passed by the temporal difference error. The error in the continuous space is shown in equation(9).

$$\delta=r+\gamma\phi^{\mathrm{T}}(x')\theta-\phi^{\mathrm{T}}(x)\theta \tag{9}$$

When a set of value functions is updated, another value function is used as a standard to evaluate the value function updated, to prevent it from getting into a local optimal because of the maximization bias. The method for updating the temporal difference of the first set of value functions is shown in equation(10).

$$\delta=r+\gamma\phi^{\mathrm{T}}(x')\theta_2-\phi^{\mathrm{T}}(x)\theta_1 \tag{10}$$

$\boldsymbol{\theta}_1$ represents the policy parameter corresponding to the first set of value functions. $\boldsymbol{\theta}_2$ represents the second one. In equation(10), the value function of the next state is computed by the second set of evaluation values, and the value function of the current state is computed by the first set of evaluation values, so that the first set of value functions will be affected by the second one when they are updated. Similarly, the temporal difference equation(11) can be updated using the second set of value functions.

$$\delta = r + \gamma\phi^{\mathrm{T}}(x')\theta_1 - \phi^{\mathrm{T}}(x)\theta_2 \quad (11)$$

In the discrete space, double Q-learning selects actions based on two sets of evaluation value functions. The common approach is to use $Q_1(u)+Q_2(u)$ instead of the original value function for action selection. However, this method is not practical for the actor-critic algorithm. In discrete space, the policy can be obtained through the state value function. But it is difficult to solve the optimal action in a continuous space only depending on the state value function.

The policy of the actor-critic algorithm is stored directly without any calculation of the value function. Typically, a policy is randomly selected. And then agent will update the corresponding set of value functions according to the selected policy. This method not only ensures that the algorithm learns according to the framework of the actor-critic, but also reduces the degree of the maximization bias. The specific actor-critic algorithm based on the cross evaluation value function is shown in algorithm 1.

**Algorithm 1.** An actor-critic algorithm based on cross evaluation of value functions

1. initialization：parameters of value function $\theta_1$， $\theta_2$， parameters of policy $\theta_1$、$\theta_2$；
2. REPEAT (for each episode)：
3. $x \leftarrow x_0$, $x_0$ is a initial state；
4. REPEAT (for each time step)：
5 perform the following steps with 50% probability：
6. Select $\theta_1$ and get the optimal action $Ac_1(x)$, and perform action $u$.
7 in state $x$, perform action $u$, get $r$ and the next state $x'$；
8. $\delta = r + \gamma\phi^{\mathrm{T}}(x')\theta_2 - \phi^{\mathrm{T}}(x)\theta_1$；
9. $\theta_1 = \theta_1 + \alpha\delta\phi(x)$；
10. 当 $\delta > 0$ 时 $\theta_1 = \theta_1 + \beta\left(u - Ac_1(x)\right)\dfrac{\partial Ac_1(x)}{\partial\theta}$；
11. With 50% probability perform the following steps：
12. From $\psi_2$ get the optimal action $Ac_2(x)$ and perform action $u$
13. In state $x$, perform the action $u$, and get a reward $r$ and the next state $x'$；
14. $\delta = r + \gamma\phi^{\mathrm{T}}(x')\theta_1 - \phi^{\mathrm{T}}(x)\theta_2$；
15. $\theta_2 = \theta_2 + \alpha\delta\phi(x)$；
16. 当 $\delta > 0$ 时 $\theta_2 = \theta_2 + \beta\left(u - Ac_2(x)\right)\dfrac{\partial Ac_2(x)}{\partial\theta}$;
17. $x = x'$；
18. UNTIL $x$ is a final state.
19. UNTIL the maximum number of episodes.

## 5. ANALYSIS OF EXPERIMENT RESULTS

Two different puddle worlds, which are easy to get into the local optimal, are used to test the performance of the DVCAC algorithm. Figure 2 is a continuous space puddle world problem. The state space is a square with a side length of 1, and two segments represent the location of the puddle. The vertex positions of the two segments are (0.1, 0.75), (0.45, 0.75) and (0.45, 0.4), (0.45, 0.8). After the agent performs an action. If the minimum distance $d$ from the puddle to the agent is less than 0.1, the reword $r$ is $-1-400*(0.1-d)$, otherwise reword $r$ is -1. In each state, the agent can move in any directions, and the distance between each move is fixed at 0.05. The moves in $x$ axis and $y$ axis are subjected to noise interference. The noise follows normal distribution with average 0 and the standard deviation 0.01. If agent after a move exceeds the bounds, it stays on the boundary. The goal of the puddle world experiment is to find

the shortest path from the start to the terminal and the path should bypass the puddles. The start position is $(0,0)$, and the terminal $(x, y)$ satisfies $x+y>1.9$. In this puddle world, there are many suboptimal solutions. Therefore, the convergence performance of the DVCAC algorithm can be profoundly examined.

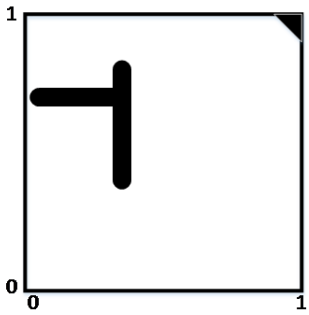



Figure 2. Puddle world 1

The DVCAC algorithm is compared with the classical CACLA algorithm [11]. In the experiment, both DVCAC and CACLA use Gaussian radial basis functions. 10x10 grids divide the whole state space. Each grid is the center of the basis function. The radius of the basis function is 0.05， discount $\gamma=0.95$, step factor of the state $\alpha=0.1$， step factor of the policy $\beta$=0.02. The maximum number of episodes is defined as 1000. The action is selected with the Gaussian exploration method. In the puddle world, the cumulative reward of each episode is used to measure the quality of the policy. The greater the return, the better the policy.

Figure 3 is the reward comparison of these two algorithms. The reward values are the average of 20 experiments. In order to prevent a large number of steps under an episode, the maximum number of steps is 100 in all episodes. Since the number of steps in each episode is very small, it is difficult to get to the goal state in an episode. The reward value can be used as a criterion for evaluating learning algorithms.

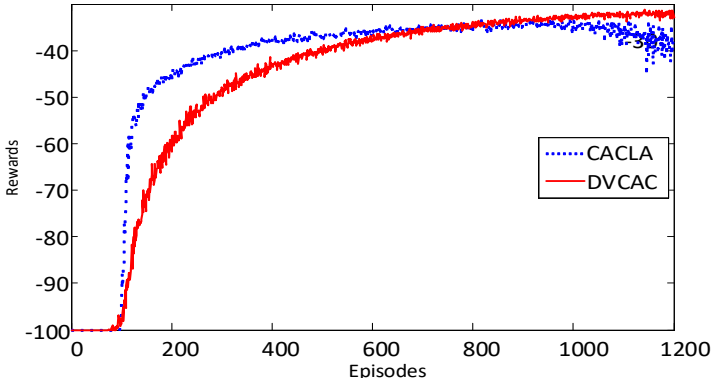



Figure 3. The comparison of accumulated rewards in puddle world 1

Since the DVCAC algorithm needs to solve and evaluate two sets of function parameters, the convergence rate is slower than that of CACLA. However, the reward value of DVCAC algorithm increases steadily with the increase of the number of episodes. When the number of episodes is about 700, the reward values of the two algorithms coincide, and the DVCAC is still optimized when the number of episodes exceeds 700. The CACLA algorithm although the learning speed is very fast, but when the number of episodes greater than 400, the reward value is almost no change, and the policy is basically not optimizing, and not stable after 1000 episodes. Experiments show that although the convergence speed is not fast, DVCAC algorithm has a good learning performance and an ideal convergence, and the learnt policy is relatively stable.

In this experiment, puddle is defined as a segment; the endpoint of the segment is (0.1, 0.5), (0.8, 0.5) respectively. If the agent goes near the puddle, and the distance is less than 0.1, the reward $r$ is defined as $-1-400*(0.1-d)$. Otherwise the reward is set to -1. In order to keep the environment more random, when the agent goes above the middle position, e.g. $y>0.5$, agent may get a random reward with mean of 0.1 and a standard deviation of 1. The task of the agent in the experiment is to reach the destination quickly. In each state the agent can move in any direction, the distance of each move is fixed to 0.05.

Each time the movement in both $x$ axis and $y$ axis will be affected by noise. The noise follows a normal distribution, with the mean value of 0, and the standard deviation of 0.01. If the movement exceeds the boundary, the agent remains on the boundary. In this experiment, the starting point is defined as $(0,0)$, the end point satisfies $x+y>1.9$. The other parameters of the puddle world 2 are exactly the same as those of the original puddle world. Puddle world 2 is shown in Figure 4. The state space is a square with a side length of 1. In this square space there is a puddle in the middle as an obstacle, blocking the movement of the agent.

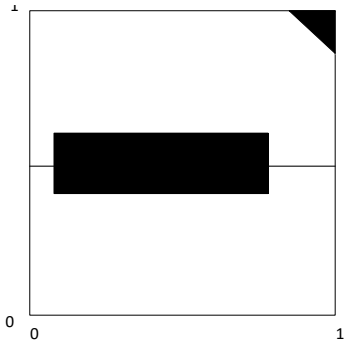



Figure 4. Puddle world 2

In the puddle world 2, the environment is simple, but there is a random noise following normal distribution in the reward part. In order to prevent excessive number of steps in an episode that eventually leads to excessive learning, the maximum number of steps defined for each episode is 100. Compared with the learning automata based, actor-cirtic algorithm in continous space, the results are shown in Figure 5.

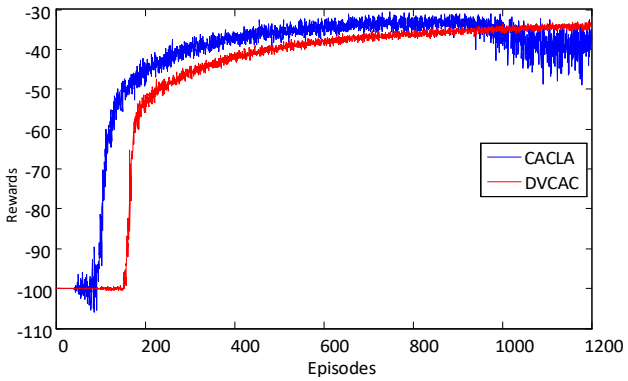



Figure 5. The comparison of accumulated rewards in puddle world 2

The reward in Figure 5 is the average of 10 experiments. Since DVCAC algorithm is a CACLA algorithm that adds a new value function to learn, the convergence rate is not as fast as CACLA. However, it can be seen from Figure 5, in DVCAC, the reward stability is much better than the CACLA algorithm. The CALCA algorithm is efficient but with a reward curve fluctuates greatly.

When the number of episodes is small (about 50 episodes), the fluctuation range is larger. In some specific circumstances, the stability of the convergence rate is most important. It can be seen from the figure that when the number of episodes is greater than 500, the stability of CACLA algorithm begins to decline, the reward starts to become smaller and there is considerable fluctuation, and DVCAC algorithm still maintains a good value-added trend. It can be seen that the DVCAC algorithm can obtain a more accurate and stable policy. The algorithm has strong robustness, and has better convergence performance, and is suitable for the application that seeks to a global optimal policy.

## 6. CONCLUSION

In order to improve the stability of the continuous space algorithm and prevent the policy from falling into the local optimal because of the lack of exploration, an actor-critic algorithm based on the cross evaluation of value function is proposed. The algorithm constructs two sets of evaluation functions, by randomly using a set of evaluation functions in the action selection. When a set of evaluation functions is updated, another set of evaluation functions is used to evaluate the value function of the next state or the next state action pair.

This method makes the policy easy get rid of the local optimal under the same exploration. The advantages are more noticeable when dealing with environments that are vulnerable to local optimal and a stable policy is required. In order to check the performance, the DVCAC and the CACLA algorithm are experimented in two different puddle world environments. The experimental results show that the policy learnt by the DVCAC algorithm is more stable than that of CACLA algorithm, and the computational complexity is of the same level.

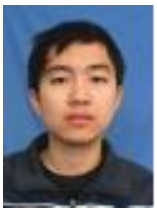
## REFERENCES

[1] Samuel A L. "Some studies in machine learning using the game of checkers". *IBM Journal of research and development 2000*, 44(1.2): 206-226.
[2] Minsky M. "Steps toward artificial intelligence". *Proceedings of the IRE*, 1961, 49(1):8-30.
[3] Barto A G, Sutton R S, Anderson C W, *et al.* "Neuronlike adaptive elements that can solve difficult learning control problems". *Transactions on systems, man, and cybernetics*, 1983, 13(5): 834-846.
[4] Konda V R, Tsitsiklis J N. "Actor-critic algorithms". *Siam journal on control & optimization, 2000,* 42(4): 1008-1014.
[5] Grondman I, Busoniu L, Lopes G A D, *et al*. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients". *Transactions on Systems Man & Cybernetics Part C, 2012,* 42(6): 1291-1307.
[6] Sutton R S, Mcallester D, Singh S, *et al.* "Policy gradient methods for reinforcement learning with function approximation", *Advances in neural information processing systems. Cambridge, MA, USA: 2000: 1057-1063.*
[7] Engel Y, Mannor S, Meir R. "Bayes meets Bellman: The gaussian process approach to temporal difference learning", *International conference on machine learning. New Jersey, USA: 2003: 154-161.*
[8] Peters J, Schaal S. "Natural Actor-critic". *Neurocomputing, 2008,* 71(7–9): 1180-1190.
[9] Peters J, Vijayakumar S, Schaal S. "Reinforcement learning for humanoid robotics". *Autonomous robot, 2003,* 12(1): 1-20.
[10] Hasselt H V. "Reinforcement learning in continuous state and action spaces*". Berlin Heidelberg: Springer, 2012.*
[11] Wierstra D, Schaul T, Peters J, *et al.* "Natural evolution strategies", *Congress on evolutionary computation. Piscataway, NJ, USA, 2008: 3381-3387.*
[12] Sun Y, Wierstra D, Schaul T, *et al.* "Efficient natural evolution strategies", *Genetic and evolutionary computation conference. Montreal, Québec, Canada, 2009: 539-546.*
[13] Rubinstein R Y, Kroese D P. "The Cross-entropy method". *New York: Springer, 2004.*
[14] Botev Z I, Kroese D P, Rubinstein R Y, *et al.* "The cross-entropy method for optimization". *Handbook of Statistics, 2013,* 31: 35-59.
[15] Martin H J A, De Lope J. "Ex<α>: An effective algorithm for continuous actions reinforcement learning problems", *Conference of the industrial electronics society. Piscataway, NJ, USA, 2009: 2063-2068.*
[16] Lillicrap T P, Hunt J J, Pritzel A, *et al.* "Continuous control with deep reinforcement learning". *Computer Science, 2015,* 8(6):A187.
[17] Gu S, Lillicrap T P, Sutskever I, *et al.* "Continuous deep Q-learning with model-based acceleration", *International conference on machine learning. New Jersey, USA, 2016: 2829-2838.*
[18] Khamassi M, Tzafestas C. "Active exploration in parameterized reinforcement learning". *arXiv preprint arXiv:1610.01986, 2016.*
[19] Bhatnagar S, Sutton R S, Ghavamzadeh M, *et al.* "Incremental natural actor-critic algorithms". *Neural information processing systems. North Miami Beach, Florida, 2007: 105-112.*
[20] Konda V R, Tsitsiklis J N. "Actor-Critic Algorithms", *Neural information processing systems. Philadelphia, PA, USA, 2000: 1008-1014.*

[21] Fu Qi-Ming, Liu Quan, Wang Hui. "A Novel of Policy Q (λ) Algorithm Based on Linerar Function Approximation". *Chinese Journal of Computers, 2014*, 37(3): 677-686. (in Chinese)
[22] Hasselt H V, Guez A, Silver D. "Deep reinforcement learning with double Q-learning". *Thirties AAAI conference on artificial intelligence.* Phoenix, USA, 2016:2094-2100.

## BIOGRAPHIES OF AUTHORS

Hui Wang, born in 1968, Ph.D. candidate. His main research interests include reinforcement learning, computer vision and human-computer interaction.



Peng Zhang, born in 1992, Master student. His main research interests include reinforcement learning in continuous spaces.

Quan Liu, born in 1969, Ph.D., professor, Ph. D. supervisor. His main research interests include reinforcement learning, intelligence information processing and automated reasoning.