2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Kalman Filter-based Actor-Critic Learning Approach

Bin Wang and Dongbin Zhao

Abstract—**Kalman filter is an efficient way to estimate the parameters of the value function in reinforcement learning. In order to solve Markov Decision Process (MDP) problems in both continuous state and action space, a new online reinforcement learning algorithm using Kalman filter technique, which is called Kalman filter-based actor-critic (KAC) learning is proposed in this paper. To implement the KAC algorithm, Cerebellar Model Articulation Controller (CMAC) neural networks are used to approximate the value function and the policy function respectively. Kalman filter is used to estimate the weights of the critic network. Two benchmark problems, namely the cart-pole balancing problem and the acrobot swing-up problem are provided to verify the effectiveness of the KAC approach. Experimental results demonstrate that the proposed KAC algorithm is more efficient than other similar algorithms.**

I. Introduction

Reinforcement learning (RL) is a powerful machine learning technique to solve the optimal control problem for a complex decision-making system. RL agent selects a control action, observes the consequences of the action and gets an immediate reward through interactions with the dynamic system. The control performance is evaluated by the expected cumulative discounted reward (namely the value function) in the long run, and the evaluation is used to update the control action so as to improve its performance. In most cases, the controlled system is modeled as a Markov Decision Process (MDP) [1], which is composed of a state set $\mathcal{S}$, an action set $\mathcal{A}$, a Markovian transition probability set $\mathcal{P}$, a reward function R and a discounting factor γ. In order to estimate the optimal policies of MDPs, a variety of value function estimation techniques [2] have been investigated in past decades in the RL community. Temporal Difference (TD) methods proposed by [3][4] are preferable value function estimation approach due to their fast convergence ability. Moreover, many algorithms based on TD methods have been studied to estimate the optimal value functions. Q-learning [5], Sarsa learning [6], Actor-Critic (AC) methods [7], and Adaptive Dynamic Programming (ADP) algorithms [8] are normally used in RL with some common characteristics, i.e., the TD error.

Most of the traditional RL methods deal with discrete state and action spaces, however, when it comes to large or continuous state and/or action spaces in real-world applications, the curse of dimensionality problem is inevitable and the control performance of the RL system cannot be guaranteed. In order to get the optimal value and policy functions for continuous state and/or action spaces, different function approximation techniques have been major focus in the research field of RL, aiming to design appropriate function approximators with high generalization capability and computational efficiency. Among which the most popular one is value function approximation (VFA) [9], and different approximation architectures such as feed-forward neural network [10], fuzzy sets [11] and kernel methods [12] are considered. Combined with TD approach, RL algorithms using VFA have been widely investigated both in theory and application in recent years [14-16]. These algorithms update parameters of the value function with rules such as gradient decent methods or least squares (LS) estimation. Least squares related algorithms such as LSTD [16][17], Kernel-based Least Squares TD (KLSTD) [18] and their eligibility variants are superior to gradient-based methods because they eliminate the design of step-size schedules and make efficient use of data. However, the computation complexity, i.e. the computation per time-step of parameter update for LSTD methods are $O(k^3)$, while recursive LS (RLS) [19] or Kalman filtering techniques can reduce this to $O(k^2)$, where k is the number of state features [16].

Kalman filter technique is our main focus in this paper. The traditional Kalman filter can be used to approximate an unknown function through a sequence of noisy samples, which is viewed as a recursive stochastic algorithm. In [20], Kalman filter is generalized to approximate the fixed point of an operator, thus the fixed point Kalman filter (FPKF) algorithm is used to produce approximate value functions in RL problems. [21] generalizes the FPKF algorithm to off-policy FPKF (λ) algorithm. [22] and [23] introduce a novel approximation scheme which is called the Kalman Temporal Differences (KTD) framework, and KTD-based algorithm is provided for deterministic MDPs, while the extended KTD (XKTD) framework is used for stochastic MDPs. In [24] Kalman filter is used to model the weights on the basis functions and a Kalman Filter Q-Learning (KFQL) method is proposed to learn an effective policy for MDP with continuous state space, and an approximate KFQL (AKFQL) by ignoring dependence among basis functions is presented as well to solve some benchmark problems in RL.

The above Kalman filter-based RL methods are the state of the art in the RL community. However, all these algorithms are about discrete action space, although some of which are involving continuous state space. Continuous state and action spaces MDPs are challenging problems arising in multiple areas of artificial intelligence. This motivates our research. In

Bin Wang and Dongbin Zhao are with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. (e-mail: bin.wang@ia.ac.cn, dongbin.zhao@ia.ac.cn).

This work was supported in part by National Natural Science Foundation of China under Grant Nos. 61273136 and 61034002, and Beijing Natural Science Foundation under Grant No. 4122083, and Visiting Professorship of Chinese Academy of Sciences.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

this paper, an online Kalman filter-based Actor-Critic (KAC) RL approach is proposed to solve both continuous state and action spaces MDPs. A parametric function approximation method, namely a neural network called Cerebellar Model Articulation Controller (CMAC) is used to approximate both the value function (the critic) and the policy function (the actor). By applying Kalman filter in the critic, parameters of the value function are updated so as to improve the evaluation performance. Then the weights of the policy function in the actor are updated with some prescribed rules to improve the control performance. The critic and the actor work alternately to learn an optimal policy online. Experimental results on some benchmark problems such as the cart-pole balancing problem and the mountain car problem are provided to verify the effectiveness of the KAC algorithm. Comparison studies with conventional actor-critic algorithm in [7] and a RLS-based actor-critic method proposed in [19] are also investigated in this paper to show the outstanding performance of the KAC algorithm.

The paper is structured in five parts. In section II, the standard Kalman filter framework is introduced. The KAC algorithm is derived and a CMAC implementation is presented in section III. Simulation experiments are provided in section IV. Section V draws some conclusions and future works are addressed.

II. KALMAN FILTER: AN OVERVIEW

Kalman filter is first proposed in 1960 by Kalman [25]. Kalman filter is generally used to estimate the state of a discrete-time controlled process. For parameter estimation problem, the parameter vector can be modeled as a random variable, which has the similar form as the state description of a controlled process. Then Kalman filter can be used to estimate the parameter vector.

The parameter vector can be stated in a state space formulation

$$\begin{cases} \hat{\theta}_{k+1} = \hat{\theta}_k, \\ \hat{\theta}_0 = \overline{\theta}_0 + \xi_0, & \xi_0 \simeq N(0, P_0) \\ y_k = z_k^T \hat{\theta}_k + e_k, & e_k \simeq N(0,1) \end{cases} \tag{1}$$

where $\hat{\theta}$ is the estimation of the parameter θ. y is an unknown function to be approximated by a linear combination of prespecified basis function z, which is similar to the output of a controlled process. The random variables ξ and e with Gaussian distribution are the process and measurement noise respectively. $\overline{\theta}_0$ and P_0 are prior knowledge of the parameter θ.

Then the Kalman filter equations are directly given as

$$\begin{cases} \hat{\theta}_{k+1} = \hat{\theta}_k + L_k (y_k - z_k^T \hat{\theta}_k), \\ L_k = \dfrac{P_k z_k}{1 + z_k^T P_k z_k}, \\ P_{k+1} = P_k - \dfrac{P_k z_k z_k^T P_k}{1 + z_k^T P_k z_k}. \end{cases} \tag{2}$$

where L_k can be viewed as Kalman gain. Details about Kalman filter derivation can be seen in [26].

Thus the parameters can be estimated in the above recursive form until to the optimal value.

In next section the parameters of the value function will be updated according to the Kalman filter equations (2), and Kalman filter-based actor-critic RL method will be derived.

III. THE KAC LEARNING ALGORITHM

In this paper we propose a Kalman filter-based actor-critic RL algorithm based on the conventional actor-critic (AC) framework [1] in RL. The structure of the AC approach is shown in Fig. 1. There are three main elements of the AC strategy, namely the *actor*, the *critic* and the *system* (also called e*nvironment*). The actor approximates the optimal policy and is used to select a control action. The critic estimates the performance of the action by a predefined value function, and a parametric approximation method is used to approximate the value function. Meanwhile, Kalman filter-based parameter estimation method is used to update the weights of the value function. The system responds to the action and transits to the next state from the current state. Details about the KAC algorithm are presented as follows.

A. The Critic

The critic approximates the value function V which is classically defined as the accumulative discounted reward value related to the state s,

$$V^{\pi}(s) = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\} \tag{3}$$

where r_t is the reward observed at discrete time t. π is a given policy, and the discounted factor γ satisfying $0 < \gamma \le 1$.

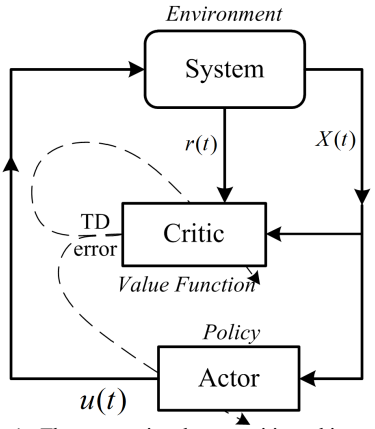

Fig. 1. The conventional actor-critic architecture.

After each action selection, the critic evaluates the performance of the action by the following TD error

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

To approximate the value function in large or continuous spaces, a classical linear function approximator is used in the critic. The value function is approximated by

$$V_i(s_t) = \phi^T(s_t) v_i \tag{5}$$

where $\phi(s_t)$ is a basis functions vector, and v is a weights vector. i is an iteration step, similarly hereinafter.

When linear approximators are used, the linear least squares estimation needs to be solved with the following cost function

$$J_i(v) = \sum_{j=1}^{i} \left\| r_j + \gamma \phi_{j+1}^T v - \phi_j^T v \right\|^2 = \sum_{j=1}^{i} \left\| r_j - \Delta\phi_j^T v \right\|^2 \tag{6}$$

where $\Delta\phi_j^T v = \phi_j^T v - \gamma\phi_{j+1}^T v$. This least squares estimation can be computed recursively with Kalman filter. Compared with (1), we can get the parameter update equation similar to (2)

$$v_i = v_{i-1} - K_i (r_i - \Delta\phi_i^T v_{i-1}) \tag{7}$$

$$K_i = \frac{P_{i-1}\Delta\phi_i}{1 + \Delta\phi_i^T P_{i-1} \Delta\phi_i} \tag{8}$$

$$P_i = P_{i-1} - \frac{P_{i-1}\Delta\phi_i \Delta\phi_i^T P_{i-1}}{1 + \Delta\phi_i^T P_{i-1} \Delta\phi_i} \tag{9}$$

Thus the critic can be implemented with a linear approximator and the parameters can be updated by equations (7) ~ (9).

B. The Actor

The actor is also implemented by a linear function approximator to approximate the control policy. Input of the actor is the current state, and the output of the actor is

$$A_i(s_t) = \psi^T(s_t) w_i \tag{10}$$

where $\psi(s_t)$ is a basis function vector, and w is the weights vector of the actor.

A noise term n_i with Gaussian probabilistic distribution is added to $A_i(s_t)$ as an exploration of the control policy. Then the actual action which directly acts on the controlled system is modified as

$$A_i'(s_t) = A_i(s_t) + n_i(0, \sigma_V(t)) \tag{11}$$

The variance of the noise is defined as [19]

$$\sigma_V(t) = \frac{k_1}{1 + \exp(k_2 V(s_t))} \tag{12}$$

where k_1 and k_2 are positive regulating constants, and $V(s_t)$ is the current value function estimation of the critic.

The actor can learn its weights through an estimation of the policy gradient [19]

$$w_i = w_{i-1} + \alpha\delta_t \frac{A_k'(s_t) - A_k(s_t)}{\sigma_V(t)} \psi(s_t) \tag{13}$$

where α is a learning rate of the actor.

In addition, the system considered in the actor-critic algorithms is just modeled to transit the system state and provide a reward signal. None of the system parameters is needed in the learning process, thus it implies that the proposed KAC algorithm is a model-free RL approach.

Algorithm 1: The KAC algorithm

Initialization

The actor weights w, learning rate α, exploration factor k_1 and k_2,

The critic weights v, and discount factor γ,

$P_0 = \beta I$.

Repeat for each trial

$s(t) \leftarrow$ initial state of trial

While $s(t)$ is not the terminal state

According to $s(t)$, compute $A(s_t)$ by (10),

and determine the actual action $A'(s_t)$ by (11)

Take action $A'(s_t)$, **observe** reward r,

Update the weights of the critic by (7)

Update the actor weights according to (13)

$s(t) \leftarrow s(t+1)$

Fig. 2. The KAC algorithm learning procedure.

The learning procedure of the KAC algorithm is shown in Fig. 2. For a given initial state, the actor outputs an action, and a noisy control action acts on the system to get the state transition and the reward. The critic generates an estimation of the value function to evaluate the control performance, and then the critic updates its weights by (7), while the actor is updated by (13). In this way the KAC algorithm iterates online until the terminal state is satisfied.

C. Neural Network Implementation

A neural network implementation of the KAC algorithm is proposed in this paper. We use a kind of feed forward neural network which is called Cerebellar Model Articulation Controller (CMAC) to approximate the actor and the critic, respectively.

CMAC is proposed by Albus in 1975 [27]. As a linear function approximator, CMAC has been widely investigated and applied in RL. Details can be seen in [27]. Fig. 3 shows the architecture of a CMAC neural network. The input state space $\boldsymbol{S}$ is mapped to the association space $\boldsymbol{A}$, and then mapped to the output space $\boldsymbol{O}$. The corresponding map functions are

$$\begin{aligned} \boldsymbol{f}_1 &: \boldsymbol{S} \Rightarrow \boldsymbol{A} \\ \boldsymbol{f}_2 &: \boldsymbol{A} \Rightarrow \boldsymbol{O} \end{aligned} \tag{14}$$

The first map $\boldsymbol{f}_1$ is a tile coding map, which detects each input state and groups the receptive fields of the features into partitions of the input state space. There are C tilings and M partitions for each input state. For multi-input state space, the total physical memory of the CMAC network is $M^n C$, with n dimension of the input. We adopt a hashing technique presented in [1] to reduce memory requirements

$$A(s) = \sum_{i=1}^{n} \left[a(i) + M^{i-1} \right] \tag{15}$$

$$F(s) = A(s) \bmod N \tag{16}$$

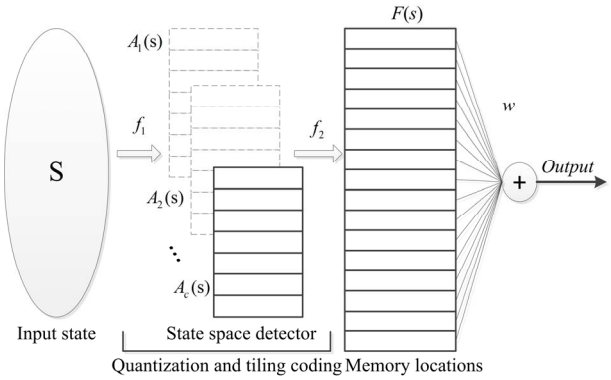

Fig. 3. The architecture of a CMAC neural network.

where $0 \le a(i) \le M$ is the activated tile for the *i*-th state, $F(s)$ is the physical memory location of the state *s*, and *N* is the number of the physical memory.

Thus the second map $\boldsymbol{f}_2$ namely the output of the CMAC network is

$$f(s) = W^T F(s) \tag{17}$$

where *W* is the adjustable weights matrix. $F(s)$ here can be viewed as the basis function vector of the actor and the critic approximation.

With this structure of CMAC network, the actor and the critic can be implemented to approximate the control policy and the value function, respectively.

IV. Experimental Results

In order to verify the effectiveness of the proposed KAC algorithm in this paper, we implement it on two benchmark problems in RL, which are the cart-pole balancing problem and the acrobot swing-up problem, respectively. Furthermore, comparison studies with conventional actor-critic algorithm [7] as well as Fast-AHC method [19] are given to investigate the data efficiency of the KAC approach.

A. The Cart-pole Balancing Problem

The cart-pole balancing problem [10] suggested by the diagram in Fig. 4 depicts the task of balancing a single pole mounted on a cart, which moves on a bounded, horizontal track. A force F is applied to the cart to keep the pole balanced and avoid out of the track boundaries. There are four states for the cart-pole system, which are θ , angle of the pole with respect to the vertical position ; $\dot{\theta}$, angle velocity; x , position of the cart on the track; and $\dot{x}$, cart velocity.

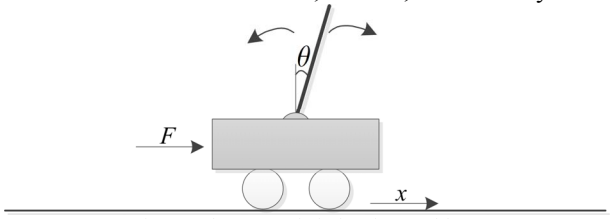

Fig. 4. The cart-pole balancing problem.

The system dynamics of the cart-pole is described by

$$\begin{cases} \ddot{\theta} = \dfrac{g\sin\theta - \cos\theta\left(F + ml\dot{\theta}^2\sin\theta + \mu_c\,\mathrm{sgn}(\dot{x})\right) - \dfrac{\mu_p\dot{\theta}}{ml}}{l\left(\dfrac{4}{3} - \dfrac{m\cos^2\theta}{m_c+m}\right)} \\ \ddot{x} = \dfrac{F + ml\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \mu_c\,\mathrm{sgn}(\dot{x})}{m_c+m} \end{cases} \tag{18}$$

The system parameters of the cart-pole are the same as [10]. The cart-pole balancing task is completed when the following state boundary constraints are satisfied

$$-5^\circ \le \theta \le +5^\circ, \quad -2.4m \le x \le +2.4m \tag{19}$$

Where the angle constraints are more severe than other related works [10][19], which can test the outstanding performance of the proposed KAC algorithm.

To learn an approximate optimal policy for the cart-pole balancing problem, the CMAC configurations of the actor and the critic neural networks need to be determined first, which are listed in Table I.

TABLE I
CMAC PARAMETERS FOR THE ACTOR AND THE CRITIC NETWORKS

Parameters	The actor network	The critic network
n	4	4
C	4	4
M	7	7
N	80	30
W	$[0,\ldots,0]_{N\times 1}$	$[0,\ldots,0]_{N\times 1}$

In addition, other initial parameters of the KAC algorithm are the actor network learning rate $\alpha = 0.5$, exploration factor $k_1 = 0.4, k_2 = 0.5$, the discount factor $\gamma = 0.95$ and $\beta = 0.04$. Online learning of the KAC algorithm for the cart-pole balancing problem begins with the state randomly initialized between $[0, 0.5]$. And according to Fig. 2, the algorithm iterates until the terminal state is reached, which satisfies (19), and then the task is completed successfully when this has lasted 12000 time steps. The reward in this problem is defined as

$$r(t) = \begin{cases} 0, & within\ bounds \\ -1, & otherwise \end{cases} \tag{20}$$

A successfully learning process is shown in Fig. 5. The angle and the position variables are all within predefined constraints. In order to investigate the Kalman filter estimation performance, we compare the KAC algorithm with other related methods. [19] proposed a Fast-AHC learning approach, which is based on recursive least squares estimation combined with actor-critic learning. Furthermore, conventional actor-critic learning algorithm is also implemented in this paper. The comparison configurations are all the same as KAC algorithm. We carry out 100 runs consisting of a maximum of 2000 consecutive trials for the cart-pole balancing problem. The comparison results are shown in Table II.

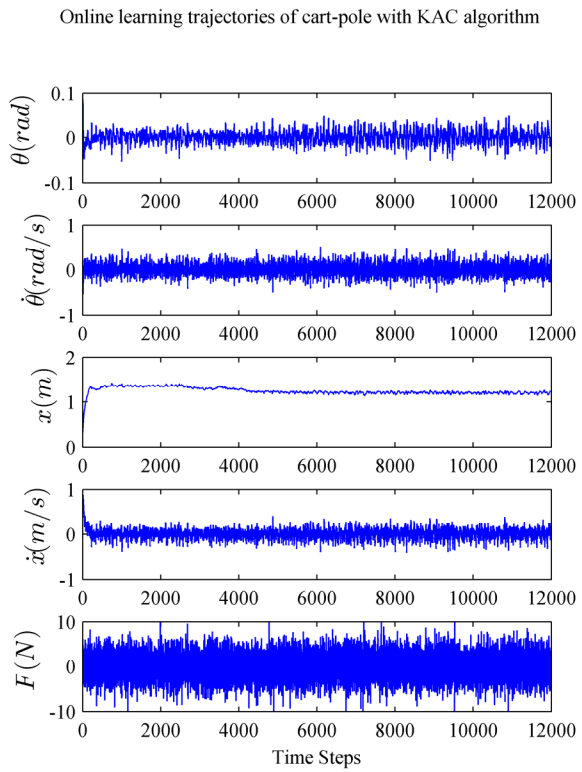

Fig. 5. Online learning process of the cart-pole balancing problem with KAC algorithm.

We compare the trials needed to balance the pole successfully. It can be concluded from Table II that the proposed KAC algorithm can complete the cart-pole balancing task with fewer trials than other algorithms, which demonstrates that the KAC algorithm presented in this paper is efficient.

TABLE II
PERFORMANCE COMPARISON RESULTS WITH OTHER METHODS

Algorithms	Minimal trials	Maximal trials	Average trials	Success rate
KAC	11	954	192	98%
Fast-AHC	12	709	217	98%
AC	10	1957	306	97%

B. The Acrobot Swing-up Problem

Acrobot [1] is a two-link, underactuated roughly analogous to a gymnast swinging on a high bar, as shown in Fig. 6. Only the second joint can exert torque. The goal of the acrobot swing-up problem is to swing the tip above line. There are four continuous state variables: two link angles θ_1 and θ_2, and two angle velocities $\dot{\theta}_1$ and $\dot{\theta}_2$. The system dynamics of the acrobot is described as follows

$$\begin{cases} \ddot{\theta}_1 = \dfrac{d_2\ddot{\theta}_2 + \phi_1}{d_1} \\ \ddot{\theta}_2 = \dfrac{\tau + d_2\phi_1/d_1 - m_2 l_1 l_{c2}\dot{\theta}_1^2 \sin\theta_2 - \phi_2}{m_2 l_{c2}^2 + I_2 - d_2^2/d_1} \end{cases} \tag{21}$$

Goal: Raise tip above line

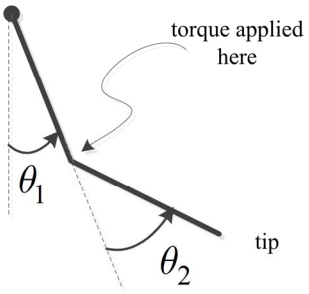

Fig. 6. The acrobot swing-up problem.

where

$$d_1 = m_1 l_{c1}^2 + m_2\left(l_1^2 + l_{c2}^2 + 2 l_1 l_{c2}\cos\theta_2\right) + I_1 + I_2$$
$$d_2 = m_2\left(l_{c2}^2 + l_1 l_{c2}\cos\theta_2\right) + I_2$$
$$\phi_1 = -m_2 l_1 l_{c2}\dot{\theta}_2^2 \sin\theta_2 - 2 m_2 l_1 l_{c2}\dot{\theta}_2\dot{\theta}_1 \sin\theta_2 + \left(m_1 l_{c1} + m_1 l_1\right) g\cos\left(\theta_1 - \pi/2\right) + \phi_2$$
$$\phi_2 = m_2 l_{c2} g\cos\left(\theta_1 + \theta_2 - \pi/2\right)$$

The parameters of the acrobot system are the same as [1]. The state variables also have bounded constraints as follows

$$\begin{aligned} &\theta_1 \in [-\pi, \pi], \quad \theta_2 \in [-\pi, \pi], \\ &\dot{\theta}_1 \in [-4\pi, 4\pi], \quad \dot{\theta}_2 \in [-9\pi, 9\pi] \end{aligned} \tag{22}$$

The acrobot swing-up problem is accepted as successful if $|\theta_1| \geq \pi$, which is the goal. Thus the reward value can be defined as

$$r(t) = \begin{cases} 100, & \textit{reach the goal} \\ -1, & \textit{otherwise} \end{cases} \tag{23}$$

To implement the KAC algorithm, we adopt the same configuration of the neural networks except that the number of the physical memory of the actor network is 100, while the number is 80 for the critic network. In addition, the learning parameters of the KAC algorithm are $\gamma = 0.9$, $\alpha = 0.2$, $\beta = 0.01$, $k_1 = 0.4$, $k_2 = 0.5$.

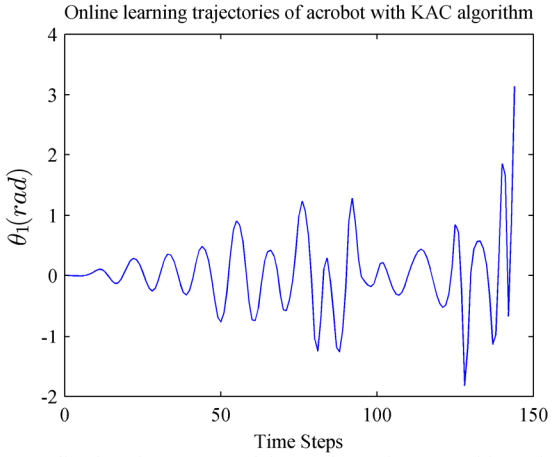

Fig. 7. Online learning process of the acrobot swing-up problem with KAC algorithm.

TABLE III
PERFORMANCE COMPARISON RESULTS WITH OTHER METHODS

Algorithms	Minimal steps	Maximal steps	Average steps	Success rate
KAC	56	290	157	99.9%
Fast-AHC	62	298	171	99.6%
AC	80	299	171	99.7%

The learning process starts from the stable equilibrium and iterates until to the goal state. The successful online learning process is shown in Fig. 7. For further comparison on the acrobot swing-up problem, we investigate the algorithms mentioned in the cart-pole balancing problem. 1000 trails which consist of a maximum of 300 time steps are carried out for each algorithm. We present the statistical results in Table III. Although these three algorithms have similarly good performance, the KAC algorithm performs better and fewer time steps are needed to swing up the acrobot. We can conclude that the KAC approach is remarkable in solving these MDPs.

V. CONCLUSION

A new reinforcement learning algorithm using Kalman filter, which is called Kalman filter-based actor critic (KAC) learning is proposed in this paper. KAC algorithm reduces the computation complexity and improves the sample efficiency, which can improve the learning performance to some extent. Kalman filter is used to estimate the parameters of the value function efficiently, and combined with actor-critic approach, the KAC algorithm can solve MDPs in both continuous state and action space. It is the first time that Kalman filter is used in RL to learn the continuous control strategy online. To implement the KAC algorithm, CMAC neural networks are used to approximate the value function and the policy function. Two benchmark problems in RL community are carried out to verify the effectiveness of the KAC approach. Experimental results demonstrate that the proposed KAC algorithm is more efficient than other similar algorithms.

There are many ways to extend the Kalman filter-based RL methods with actor-critic structures. Value function used in this paper is a state value function, which can be easily extended to state-action value function (or Q-function) with little modification. Furthermore, the theoretical proof deserves to be researched in the future.

ACKNOWLEDGMENT

The authors would like to thank Dr. Xin Xu for his publicly available simulation codes for Fast-AHC and cart-pole balancing learning.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge: MIT press, 1998.

[2] L. Busoniu, R. Babuska, B. De Schutter and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010.

[3] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9-44, 1988.

[4] P. Dayan, "The convergence of TD(λ) for general λ ," *Mach. Learn.*, vol. 8, pp. 341-362, 1992.

[5] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279-292, 1992.

[6] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Technical Report CUED/F-INFENG/TR166, Engineering Department, Cambridge University, UK, 1994. Available: http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf

[7] V. R. Konda and J. Tsitsiklis, "Actor-critic algorithms," In *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference*, Denver, Colorado, 2000, pp. 1008–1014.

[8] J. J. Murray, C. J. Cox, G. G. Lendaris and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Sys. Man Cybern. Part C*, vol. 32, no. 2, pp. 140-152, 2002.

[9] M. Geist and O. Pietquin, "Algorithmic survey of parametric value function approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol.24, no.6, pp.845,867, June 2013

[10] J. Si, Y. T. Wang, "On-line learning control by association and reinforcement," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 264-276, 2001.

[11] D. B. Zhao, Y. H. Zhu, H. B. He, "Neural and fuzzy dynamic programming for under-actuated systems," in *Proc. 2012 IEEE Int. Joint Conf. Neural Networks (IJCNN 2012)*, June 10-15, 2012, pp. 1895-1900.

[12] G. Taylor and R. Parr, "Kernelized value function approximation for reinforcement learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, New York, 2009, pp. 1017-1024.

[13] B. Wang, D. Zhao, C. Alippi and D. Liu, "Dual heuristic dynamic programming for nonlinear discrete-time uncertain systems with state delay," *Neurocomputing*, 3 August 2013. Available: http://dx.doi.org/10.1016/j.neucom.2013.06.037

[14] D. Zhao, B. Wang and D. Liu, "A supervised Actor–Critic approach for adaptive cruise control," *Soft Comput.*, vol. 17, no. 11, pp. 2089-2099, 2013.

[15] Y. Zhu and D. Zhao, "Online model-free RLSPI algorithm for nonlinear discrete-time non-affine systems." in *Neural Information Processing (ICONIP 2013)*, Springer Berlin Heidelberg, 2013, pp. 242-249.

[16] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, no. 2-3, pp. 233–246, 2002.

[17] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107-1149, 2003.

[18] X. Xu, D. Hu and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol.18, no.4, pp. 973-992, July 2007.

[19] X. Xu, H.G. He and D.W. Hu, "Efficient reinforcement learning using recursive least-squares methods," *J. Artif. Intell. Res.*, vol. 16, pp. 259-292, 2002.

[20] D. Choi and B. V. Roy, "A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning," *Discret. Event Dyn. Syst.*, vol. 16, no. 2, pp. 207-239, 2006.

[21] M. Geist and B. Scherrer, "Off-policy Learning with Eligibility Traces: A Survey", *J. Mach. Learn. Res.*, vol. 15, pp. 289-333, 2014.

[22] M. Geist, O. Pietquin and G. Fricout, "Kalman temporal differences: the deterministic case," *IEEE Symp. Adapt. Dyn. Program. Reinf. Learn. (ADPRL '09)*, 2009, pp. 185-192.

[23] M. Geist and O. Pietquin, "Kalman temporal differences," *J. Artif. Intell. Res.*, vol. 39, no. 1, pp. 483-532, 2010.

[24] C. Tripp and R. Shachter, "Approximate Kalman filter Q-learning for continuous state-space MDPs," in *Proc. 29th Conf. Uncertain. Artifi. Intell.*, Bellevue, 2013, pp. 644-653.

[25] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no.1, pp. 35-45, 960.

[26] G. Welch and G. Bishop, "An introduction to the Kalman filter," 1995. Available: http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html

[27] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)." *J. Dyn. Syst. Measur. Control*, vol. 97, no. 3, pp. 220-227, 1975.