Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Model-free Adaptive Dynamic Programming for Optimal Control of Discrete-time Affine Nonlinear System [⋆]

Zhongpu Xia [*] Dongbin Zhao [*] Huajin Tang [**]

[*] *The State Key Laboratory of Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: zhongpu.xia@gmail.com, dongbin.zhao@ia.ac.cn).*
[**] *Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR) Singapore 138632, Email: htang@i2r.a-star.edu.sg*

Abstract: In this paper, a model-free and effective approach is proposed to solve infinite horizon optimal control problem for affine nonlinear systems based on adaptive dynamic programming technique. The developed approach, referred to as the actor-critic structure, employs two multilayer perceptron neural networks to approximate the state-action value function and the control policy, respectively. It uses data collected arbitrarily from any reasonable sampling distribution for policy iteration. In the policy evaluation phase, a novel objective function is defined for updating the critic network, and thus makes the critic network converge to the Bellman equation directly rather than iteratively. In the policy improvement phase, the action network is updated to minimize the outputs of the critic network. The two phases alternate until no more improvement of the control policy is observed, such that the optimal control policy is achieved. Two simulation examples are provided to show the effectiveness of the approach.

Keywords: Model-free adaptive dynamic programming, reinforcement learning, policy iteration, multilayer perceptron neural network.

1. INTRODUCTION

Steering systems from one state to another by the application of a series of control actions in a best way is desired in the practical engineering. The one which can satisfy this goal is called the optimal control policy. It is well known that deriving optimal control policies for linear systems is equivalent to calculate the unique positive definite solution of the algebraic Riccati equation (ARE), and for nonlinear systems is to calculate the Hamilton-Jacobi-Bellman (HJB) equation(Lewis et al. (2012)). What is indispensable during the calculation is an accurate dynamics model of the system. Unfortunately, dynamics models are always completely unknown or their parameters are uncertain in the engineering. Moreover, the solution to HJB equation is often difficult or impossible to obtain as it involves solving the nonlinear partial difference equations. Therefore, developing an algorithm that can achieve the optimal control policy model-freely is fairly necessary.

Reinforcement learning makes it possible to obtain a viable control policy model-freely as the control policy can be improved by interacting with the system. More in detail, it can learn to make a good decision by observing its own behaviors and the system's responses, and use built-in mechanism for improving the control policy through a reinforcement scheme (Bertsekas and Tsitsiklis (1995)). Lots of work has been done to yield a viable control policy without accessing to the dynamics model in the computational intelligence community such as Q-learning (Watkins (1989)), the fitted Q-iteration (Ernst et al. (2005)), the least square policy iteration (LSPI) (Lagoudakis and Parr (2003)), and in the control community such as the adaptive dynamic programming (ADP) (Wang et al. (2009)). The first two algorithms focus on discrete state-action space, and the LSPI algorithm mainly focuses on continuous state but discrete action space. However, events are always continuous in the application. In order to steer the system in an optimal way, the continuous action is required. The ADP algorithm is available for the such a problem as it can calculate the numerical solution to the HJB equation.

The ADP technique has gained much interest so far in the area of deriving optimal control policy without accessing the dynamics model. It derived the optimal control policy for the linear system without accessing to the dynamics model (Bradtke et al. (1994)) and was applied for adaptive cruise control system (Zhao and Xia (2013)). Recently, the ADP technique has been studied to derive the optimal control solutions for nonlinear systems. The optimal control policy was derived for the affine nonlinear system with known partially dynamics model (Dierks and Jagannathan (2012); Vrabie and Lewis (2009)), i.e. only the control coefficient matrix was required. Model-free methods for the nonlinear system were developed in (Wang et al. (2012);

[⋆] This work is supported by National Natural Science Foundation of China (NSFC) under Grants No. 61273136, No. 61034002, and Beijing National Science Foundation under grant No. 4122083.

Zhang et al. (2009)), but the dynamics model was still required to be identified based on the collected data in advance. The action dependent ADP algorithm proposed in(Liu et al. (2001); Si and Wang (2001); Zhao et al. (2014)) provides a good mechanism for deriving the optimal control policy model-freely by using the state-action value function, but an effective way for implementing this algorithm is still being explored.

In this paper, we propose a fast and effective approach for implementing the action dependent ADP algorithm by introducing the learning mechanism in the LSPI algorithm which is efficient in using of data. Different from the LSPI algorithm, this paper focuses on the optimal control problem in the continuous state-action domain, and the multilayer preceptron (MLP) neural network which has a more powerful generalization than the linear approximation architecture (Lagoudakis and Parr (2003)) is employed to implement the actor-critic structure. The policy iteration method is adopted for calculate the numerical optimal control solution. In the policy evaluation phase, a novel objective function is defined for updating the critic network to satisfy the Bellman equation directly. In the policy improvement phase, the action network is updated to minimize the corresponding output of the critic network. What is required for implementing the policy iteration is a batch of data about the state transition of the system, and the same data can be used throughout the iteration. In addition, the Levenberg-Marquardt method is adopted to train both the critic network and the action network.

This paper is organized as follows. The problem is introduced in section 2. Section 3 outlines the proposed approach and implements it by MLP neural networks. Two affine nonlinear systems are given to verify its effectiveness in section 4. Conclusions are given in section 5.

2. PROBLEM STATEMENT

In this paper, we study the discrete-time affine nonlinear systems as follows:

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{1}$$

where $x \in \mathbb{S} \subseteq \mathbb{R}^n$ is the state vector, $u \in \mathbb{A} \subseteq \mathbb{R}^m$ is the control action vector, with the subscript $k = 0, 1, 2, \cdots$ represents the time step, $\mathbb{S}$ and $\mathbb{A}$ represent the state space and the action space respectively. It is supposed that the system is controllable and $(0, 0)$ is the equilibrium point. A control policy is a mapping from a state to an action, and denoted as π.

A value function is defined to evaluate the performance of control policy π by summing a local cost $c(x_k, u_k)$ of the system following the control policy π in infinite horizon

$$J^\pi(x_k) = \sum_{i=0}^{\infty} \gamma^i c(x_{k+i}, \pi(x_{k+i})) \tag{2}$$

where γ $(0 < \gamma \leq 1)$ is the discount factor, and the local cost $c(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$ ($Q \in \mathbb{R}^{n*n}$ is positive semi-definite matrix and $R \in \mathbb{R}^{m*m}$ positive definite matrix). At the equilibrium point, $J(0) = 0$.

Definition 1. (Admissible control policy). A control policy π is said to be admissible, if $\pi(0) = 0$, $\pi(x_k)$ is continuous in the state space $\mathbb{S}$ and it is able to drive the system to the equilibrium point from any initial state with the value function being finite (Wang et al. (2012)).

Mostly, the optimal control policy is desired, which can obtain an minimized value function

$$J^*(x_k) = \min_{\pi} \sum_{i=0}^{\infty} \gamma^i c(x_{k+i}, \pi(x_{k+i})). \tag{3}$$

According to Bellman's principle of optimality, the optimal value function is time-invariant and satisfies the discrete time HJB equation

$$J^*(x_k) = \min_{u_k}\{c(x_k, u_k) + \gamma J^*(x_{k+1})\}. \tag{4}$$

Then the optimal control action u_k^* is derived by applying the stationary condition $\frac{\partial J^*(x_k)}{\partial u_k} = 0$. Combining with the system dynamics (1), we get:

$$u_k^* = -\frac{1}{2}\gamma R^{-1} g^T(x_k) \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}}. \tag{5}$$

From (5), the optimal control action can be calculated only when the future state is acquired. It means that the dynamics model (1) must be known during the calculation of the optimal control policy. However, the dynamics model is always unknown or uncertain in the application. Although the dynamics model is set up, the solution to (5) for most nonlinear systems is difficult or even impossible to calculate. To circumvent these deficiencies, a new approach, the action dependent adaptive dynamic programming algorithm, is introduced to derive the optimal control policy for affine nonlinear system model-freely.

3. MODEL-FREE ADP

The value function defined previously gives an evaluation for each state only. The Q-function proposed by Watkins (Watkins (1989)) gives an evaluation for each state-action pair. It is also known as state-action value function (Sutton and Barto (1998)) and defined as

$$Q^\pi(x_k, u_k) = c(x_k, u_k) + \gamma J^\pi(x_{k+1}) \tag{6}$$

The state-action value function $Q^\pi(x_k, u_k)$ is the sum of the current step local cost incurred by taking action u_k from state x_k, plus the value function of next state x_{k+1} following control policy π. u_k is not specified by the control policy for x_k, and it can be any one control action in the action space $\mathbb{A}$ for that state. Thus, $Q^\pi(x_k, u_k)$ gives an evaluations for all control actions on each state with following a control policy π. Then a better action u'_k for a state x_k can be obtained directly through minimizing the state-action value function:

$$u'_k = \arg\min_{u_k \in \mathbb{A}} Q^\pi(x_k, u_k). \tag{7}$$

Remark 2. In order to derive the improved control policy π_{i+1}, the minimum solution of $Q^{\pi_i}(x_k, u_k)$ with respect to u_k should be derived. As $\frac{\partial^2 Q(x_k,u_k)}{\partial u_k^2} = R$, R is positive definite. Thus, $Q(x_k, u_k)$ is a convex function with respect to u_k in the global action space. That is to say, the solution to (7) is the global minimum value.

The policy iteration method is always used for deriving the optimal control policy. Providing an initial admissible control policy π_0, its state-action value function (6) is determined, then a new better control policy is yielded according to (7). This procedure repeats until no improvement of the control policy is observed. It can be depicted as:

- Policy evaluation:
$$Q^{\pi_i}(x_k, u_k) = c(x_k, u_k) + \gamma Q^{\pi_i}(x_{k+1}, \pi_i(x_{k+1})). \quad (8)$$
- Policy improvement:
$$\pi_{i+1}(x_k) = \arg\min_{u_k \in \mathbb{A}} Q^{\pi_i}(x_k, u_k). \quad (9)$$

Where i represents the $i-th$ policy iteration.

3.1 Framework

In order to implement the policy iteration method, the actor-critic structure is employed, where the critic is updated in the policy evaluation phase and the actor is updated in the policy improvement phase. Artificial neural network provides a good solution for the update of the critic-actor for the properties of adaptivity, self-learning and generalization. Here we employ two three-layer neural networks to approximate the state-action value function and the control policy, named critic network and action network, respectively. The input of the critic network is a row vector including states and actions, and the output is the estimation of $Q(x_k, u_k)$:

$$\hat{Q}(x_k, u_k) = w_2^T \tanh(w_1^T [x_k^T, u_k^T]^T + b_1) + b_2. \quad (10)$$

The input of the action network is a state vector and the output is the estimation of $\pi(x_k)$:

$$\hat{\pi}(x_k) = \tanh(v_2^T \tanh(v_1^T x_k + a_1) + a_2. \quad (11)$$

Where $\tanh(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$ is the activation function. w, v are the weights of the neural networks and a, b the bias, with the subscript representing the located layer. Note that, the output of the action network is limited in the range $[-1, 1]$. It needs to be transformed into the real action space in proportion when acting on the system.

For simplicity, we collect the critic network parameters w_1, w_2, b_1, b_2 into a row vector ω_c and the action network parameters v_1, v_2, a_1, a_2 into a row vector ω_a. Thus the critic network and the action network are denoted:

$$\begin{cases} \hat{Q}(x_k, u_k) = \Phi(x_k, u_k, \omega_c) \\ \hat{\pi}(x_k) = \Psi(x_k, \omega_a). \end{cases} \quad (12)$$

where $\Phi(\cdot)$ and $\Psi(\cdot)$ is the function expression for the critic network and the action network, respectively.

3.2 Critic Network

The critic network is updated to obtain the state-action value function for current control policy in the policy evaluation phase. Notice that, the state-action value function (6) is an implicit expression. The forward-in-time and the backward-in-time (Wang et al. (2009)) approaches are always used to calculate the state-action value function iteratively (Si and Wang (2001); Liu et al. (2001)). In other words, the output of the current critic network plus the local cost is chosen as the target output of the critic network during the next update iteration, and this procedure repeats until the critic network converges. As the approximation error cannot be eliminated in the neural networks, errors will accumulate throughout the iterations. Thus it will cost lots of time or be hard to converge. Moreover, the errors will be propagated to and amplified in the action network, and then lead to a failure action network. In order to avoid the accumulation of the approximation errors, a novel approach is developed in this paper, which can train the critic network to achieve the Bellman equation directly rather than iteratively.

From (8), we notice that the local cost is the relative value of state-action value function between two consecutive states. The value functions at all state-action pairs can be calculated combining with absolute value function at equilibrium point, $Q(0, 0) = 0$, if all tuples $(x_k, u_k, c(x_k, u_k), x_{k+1})$ distributed in the state-action space are collected.

Benefited from the generalization properties of the neural network, we can calculate the approximate state-action value function with a certain amount of such tuples instead of all the state-action pairs in the space. What is required is that x_k and u_k should distribute in the state-action space randomly. In addition, these tuples can be collected from the running trajectories of the system if the dynamics model is unknown.

Motivated by this idea, the update of the critic network can be depicted by an optimization problem.

$$\begin{cases} \min & \omega_c^T \omega_c \\ st. & \Phi(x_k, u_k, \omega_c) = c(x_k, u_k) + \gamma \Phi(x_{k+1}, \hat{\pi}(x_{k+1}), \omega_c). \\ & \Phi(0, 0, \omega_c) = 0 \end{cases} \quad (13)$$

Here the norm of ω_c is minimized for smoothing the outputs of the critic network. Further on, it can be changed into an objective function E_c for updating the parameters of the critic network by introducing the relative error e_r, the equilibrium point error e_0 and the parameters norm e_{ω_c} as follows:

$$\begin{cases} \min \; E_c = \frac{1}{2}(e_r^T e_r + \lambda_0^2 e_0^T e_0 + \lambda_{\omega_c}^2 e_{\omega_c}^T e_{\omega_c}) \\ e_r = \Phi(x_k, u_k, \omega_c) - \gamma \Phi(x_{k+1}, \hat{\pi}(x_{k+1}), \omega_c) - c(x_k, u_k) \\ e_0 = \Phi(0, 0, \omega_c) \\ e_{\omega_c} = \omega_c^T \omega_c \end{cases} \quad (14)$$

where λ_0, λ_{ω_c} are the weights, which can be adjusted to trade off e_r, e_0 and e_{ω_c} during the update of the parameters. λ_0 should be set to guarantee that the output at equilibrium point approximates zero as much as possible, and λ_{ω_c} is always set small as e_{ω_c} are less important than e_r and e_0 . The parameters are trained to minimize E_c and the Levenberge-Marquardt method is employed:

$$\omega_c(t+1) = \omega_c(t) - \frac{\mathbf{J}_c^T}{\mathbf{J}_c^T\mathbf{J}_c + \mu_c\mathbf{I}} e_c \tag{15}$$

where t is the inner-loop cycle for updating the parameters and μ_c is the damping factor which can be adjusted according to the variation of the objective function at each inner-loop cycle. $e_c = [e_r^T,\ \lambda_0 e_0^T,\ \lambda_{\omega_c} e_{\omega_c}^T]^T$, $\mathbf{I}$ the identity matrix with suitable dimensions and $\mathbf{J}_c$ is the Jacobian matrix of e_c with respect to ω_c

$$\mathbf{J}_c = \frac{\partial e_c}{\partial \omega_c^T} = [\frac{\partial e_r^T}{\partial \omega_c}, \lambda_0 \frac{\partial e_0^T}{\partial \omega_c}, \lambda_{\omega_c}\frac{\partial e_{\omega_c}^T}{\partial \omega_c}]^T. \tag{16}$$

After the critic network is convergent, it goes into the policy improvement phase.

3.3 Action Network

In the policy improvement phase, the action network aims to minimize the output of the critic network. The target control action is

$$\hat{\pi}'(x_k) = \arg\min_{u_k\in\mathbb{A}} \Phi(x_k, u_k, \omega_c). \tag{17}$$

Combining (17) with (12), the problem of updating the parameters of the action network to target control actions is converted into an optimization problem by minimizing the output of the critic network with respect to ω_a. Then the parameters of the action network are updated to minimize the objective function E_a as follows:

$$\begin{cases} E_a = \frac{1}{2}(e_{c_a}^T e_{c_a} + \lambda_{\omega_a}^2 e_{\omega_a}^T e_{\omega_a}) \\ e_{c_a} = \Phi(x_k, \Psi(x_k, \omega_a), \omega_c) \\ e_{\omega_a} = \omega_a^T \omega_a \end{cases} \tag{18}$$

where λ_{ω_a} is the weight of the norm of the parameters. It is adjusted to smooth the output of the action network. Similar to the update of the critic network, the parameters of the action network are updated:

$$\omega_a(t+1) = \omega_a(t) - \frac{\mathbf{J}_a^T}{\mathbf{J}_a^T\mathbf{J}_a + \mu_a\mathbf{I}} e_a^T \tag{19}$$

where t is the inner-loop cycle for updating the parameters and μ_a is the damping factor, $e_a = [e_{c_a}^T,\ \lambda_{\omega_a} e_{\omega_a}^T]^T$ and $\mathbf{J}_a$ is the Jacobian matrix of e_a with respect to ω_a.

4. SIMULATION RESULTS

In this section, the proposed approach is tested on the discrete-time nonlinear mass-spring system and the continuous-time nonlinear oscillator system, respectively.

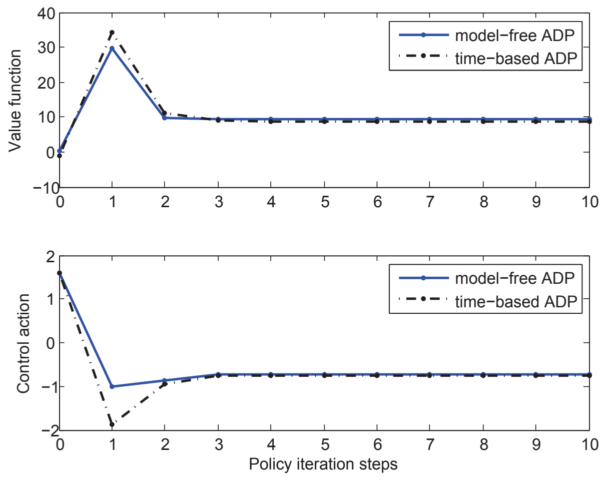

Fig. 1. The convergence process for the state $[-0.8,\ 0.8]^T$.

In the first simulation example, we compare it with an ADP method which derives the optimal control policy with the partial knowledge of the system model. In the second example, the approach is tested in the condition of different initial admissible control policies.

4.1 Discrete-time Mass-Spring System

We consider the mass-spring system(Zhang et al. (2009)) whose dynamics is given by:

$$\begin{cases} x_{1,k+1} = 0.05x_{2,k} + x_{1,k} \\ x_{2,k+1} = -0.0005x_{1,k} - 0.0335x_{1,k}^3 \\ \qquad\qquad +x_{2,k} + 0.05u_k \end{cases} \tag{20}$$

The parameters of the quadratic cost function are chosen as $Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ and $R = 0.25$. Here two three-layer perceptron neural networks are employed as the critic network and the action network with structures of $3-8-1$ and $2-6-1$, respectively. The initial parameters for the networks are chosen randomly in the range $[-1,\ 1]$. The errors' weights are set as $\lambda_0 = 10$, $\lambda_{\omega_c} = 0.01$, $\lambda_{\omega_a} = 0.01$. The initial admissible control policy is $\pi_0(x_k) = [-3,\ -1]x_k$ and discount factor $\gamma = 1$. We define the state space as $-1 \le x \le 1$. If the state space is larger than this, preprocess will be introduced to normalize states in the range $[-1,\ 1]$ in proportion when input to the both networks. It is assumed that a batch of 1000 states x_k distributed randomly in the state space, the actions for the states as $u_k = \pi_0(x_k) + N(0, \sigma^2)$, and the next time step states x_{k+1} have been collected in advance. Here we set $\sigma = 0.3$ to cover the action space randomly. These sampled data is used throughout the policy iteration.

Closed form solution of the HJB equation is difficult to yield, and no benchmarking method exists currently for evaluating the optimal control policy. The approach proposed in (Dierks and Jagannathan (2012)), which is named as time-based ADP, derives the optimal control policy with only accessing to the control coefficient matrix. It is also applied to the system (20), where the bias function of the critic is $\{x_1^2, x_1x_2, x_2^2, x_1^4, x_1^3x_2, \cdots, x_2^6\}$. The same action network and the same initial control policy are used for comparing.

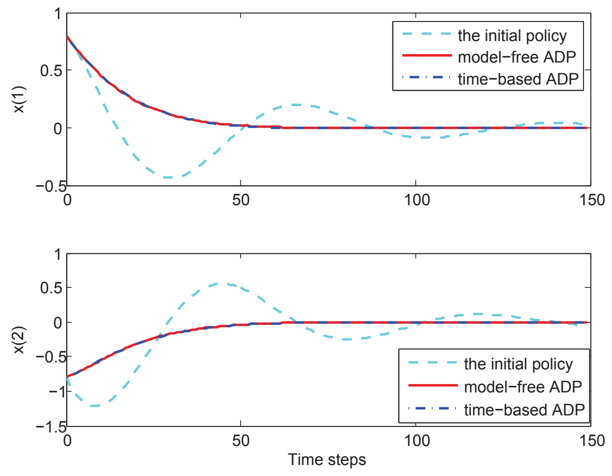

Fig. 2. The state trajectories from the state $[0.8,\ -0.8]^T$.

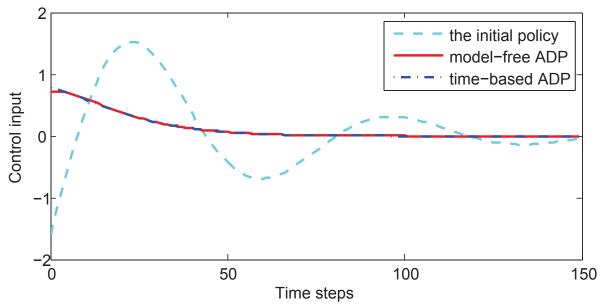

Fig. 3. The action trajectories from the state $[0.8,\ -0.8]^T$.

Both approaches are trained offline. In the training procedure, the policy iteration lasts 10 steps. Fig. 1 illustrates the convergence processes of the value function and the control action on $x = [-0.8,\ 0.8]^T$, which is derived by the model-free ADP and time-based ADP. At the $0 - th$ iteration, the value function is uncertain since the parameters of the critic networks are generated randomly. The value function is calculated for the initial control policy at the $1 - st$ iteration, and then decreases as the improvement of control policy. After the $3 - rd$ policy iteration, the value function and the control action are steady, which is deemed convergent. The value functions calculated by the both approaches converge to approximately equal value. It is the same for the control actions.

Fig. 2 shows the state trajectories from the state $[0.8,\ -0.8]^T$ to the equilibrium point steered by the initial control policy, the control policy derived by time-based ADP and the one derived by the proposed model-free ADP, and Fig. 3 shows the corresponding control action trajectories. All these trajectories are very close to the ones yielded by time-based ADP. It is also shown that both the yielded optimal policies are much better than the initial control policy, with less response time and smoother trajectories.

All these results illustrate the proposed approach has the same optimization ability with the time-based ADP approach, but accessing to no knowledge of the dynamics model.

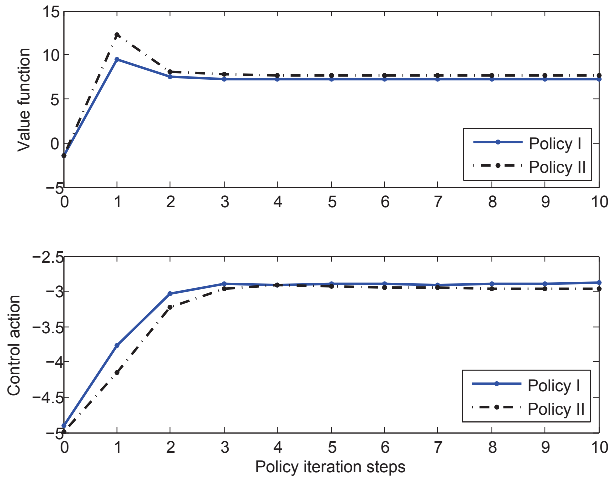

Fig. 4. The convergence process for the state $[0.5,\ 0.8]^T$.

4.2 Continuous-time Oscillator System

The continuous-time oscillator system (Abu-Khalaf and Lewis (2005)) is considered with dynamics as follows:

$$\begin{cases} \dot{x_1} = x_1 + x_2 - x_1(x_1^2 + x_2^2) \\ \dot{x_2} = -x_1 + x_2 - x_2(x_1^2 + x_2^2) + u. \end{cases} \tag{21}$$

It is difficult to derive the discrete-time control coefficient matrix of (21), so time-based ADP is unavailable for such a system. The proposed model-free approach would be carried out on the system. The sample time is set as $0.1s$.

The quadratic cost function is chosen as $Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ and $R = 0.1$. As this system is more complex than the previous system, the number of hidden neurons is increased to 10 and 8 for the critic network and the action network, respectively. Two different initial admissible control policies: Policy I ($\pi_0(x_k) = [-5,\ -3]x_k$) and Policy II ($\pi_0(x_k) = [-6,\ -2.5]x_k$), are provided for the policy iteration. Other parameters and the way of sampling data are same as the settings of the previous example.

In the training procedure, the policy iteration lasts 10 steps and Fig. 4 illustrates the convergence processes of the value function and the action on the states $[0.5,\ 0.8]^T$ from the different initial control policies, respectively. The value function decreases following the improvement of the control policies. Both the value function and the control action converge at the $3 - rd$ iteration. The state trajectories from an initial state $[-0.8,\ 0.8]^T$ to the equilibrium point steered by the two initial control policies, the optimal control policy yielded from Policy I and the one yielded from Policy II are shown in Fig. 5, and the corresponding control action trajectories are displayed in Fig. 6. Although the trajectories steered the initial control policies are fluctuant, the ones steered by yielded control policies are smooth and reach the equilibrium point fast. Moreover, the trajectories steered by yielded control policies are very close to each other.

These results illustrate that the proposed approach is able to obtain almost the same control policy even if different initial admissible control policies are provided.

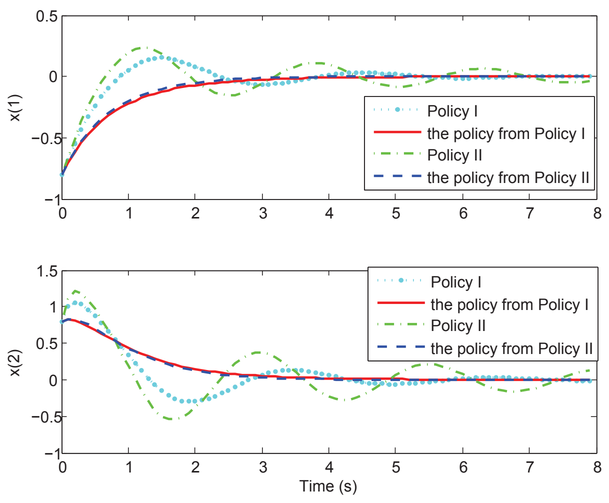

Fig. 5. The state trajectories from the state $[-0.8,\ 0.8]^T$.

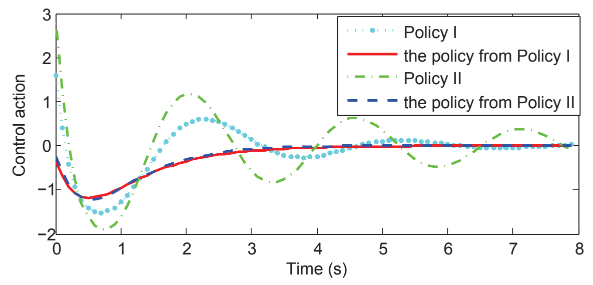

Fig. 6. The action trajectories from the state $[-0.8,\ 0.8]^T$.

5. CONCLUSIONS

In this paper, a model-free approach is proposed to achieve the optimal control policy for discrete-time affine nonlinear system based on the action dependent ADP technique. The actor-critic structure is adopted, which is implemented by two MLP neural networks. The critic network is trained in a novel way to approximate the Bellman equation directly in the policy evaluation phase. Both the critic network and the action network are trained by Levenberg-Marquardt method. Two simulation examples are given to verify the effectiveness of the proposed approach. The first simulation example illustrates that the proposed model-free ADP is effective as same as time-based ADP where partial model of the system is required. The second simulation example illustrates that the model-free ADP is able to achieve the optimal control policy although different initial control policies are provided. In both examples, the control policy converges after the policies iterates three times. Moreover, only the number of the hidden neurons need be adjusted to yield a more powerful generalization ability for systems with varying degrees of complexity. All results show the proposed approach is effective, adaptable and fast learning.

Further work will be on the analysis the admissibility of the improved control policy and give a convergence proof for the proposed approach.

REFERENCES

Abu-Khalaf, M. and Lewis, F.L. (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica*, 41(5), 779–791.

Bertsekas, D.P. and Tsitsiklis, J.N. (1995). Neuro-dynamic programming: An overview. In *Proceedings of IEEE Conference on Decision and Control*, volume 1, 560–564.

Bradtke, S.J., Ydstie, B.E., and Barto, A.G. (1994). Adaptive linear quadratic control using policy iteration. In *Proceedings of American Control Conference*, volume 3, 3475–3479.

Dierks, T. and Jagannathan, S. (2012). Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7), 1118–1129.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *The Journal of Machine Learning Research*, 6, 503–556.

Lagoudakis, M.G. and Parr, R. (2003). Least-squares policy iteration. *The Journal of Machine Learning Research*, 4, 1107–1149.

Lewis, F.L., Vrabie, D.L., and Syrmos, V.L. (2012). *Optimal Control*. John Wiley & Sons, Inc., 3rd edition.

Liu, D., Xiong, X., and Zhang, Y. (2001). Action-dependent adaptive critic designs. In *Proceedings of International Joint Conference on Neural Networks*, 990–995.

Si, J. and Wang, Y.T. (2001). Online learning control by association and reinforcement. *IEEE Transactions on Neural Networks*, 12(2), 264–276.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement learning: An introduction*. 1. Cambridge Univ Press.

Vrabie, D. and Lewis, F. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks*, 22(3), 237–246.

Wang, D., Liu, D., Wei, Q., Zhao, D., and Jin, N. (2012). Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica*, 48(8), 1825–1832.

Wang, F.Y., Zhang, H., and Liu, D. (2009). Adaptive dynamic programming: an introduction. *IEEE Computational Intelligence Magazine*, 4(2), 39–47.

Watkins, C.J.C.H. (1989). *Learning from delayed rewards*. Ph.D. thesis, University of Cambridge.

Zhang, H., Luo, Y., and Liu, D. (2009). Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Transactions on Neural Networks*, 20(9), 1490–1503.

Zhao, D., Hu, Z., Xia, Z., Alippi, C., Zhu, Y., and Wang, D. (2014). Full-range adaptive cruise control based on supervised adaptive dynamic programming. *Neurocomputing*, 125, 57–67.

Zhao, D. and Xia, Z. (2013). Adaptive optimal control for the uncertain driving habit problem in adaptive cruise control system. In *Proceedings of IEEE International Conference on Vehicular Electronics and Safety*, 159–164.