Soft Computing
https://doi.org/10.1007/s00500-018-3063-7

METHODOLOGIES AND APPLICATION

Adaptive cruise control via adaptive dynamic programming with experience replay

Bin Wang[1,2] · **Dongbin Zhao**[2] · **Jin Cheng**[1]

Abstract
The adaptive cruise control (ACC) problem can be transformed to an optimal tracking control problem for complex nonlinear systems. In this paper, a novel highly efficient model-free adaptive dynamic programming (ADP) approach with experience replay technology is proposed to design the ACC controller. Experience replay increases the data efficiency by recording the available driving data and repeatedly presenting them to the learning procedure of the acceleration controller in the ACC system. The learning framework that combines ADP with experience replay is described in detail. The distinguishing feature of the algorithm is that when estimating parameters of the critic network and the actor network with gradient rules, the gradients of historical data and current data are used to update parameters concurrently. It is proved with Lyapunov theory that the weight estimation errors of the actor network and the critic network are uniformly ultimately bounded under the novel weight update rules. The learning performance of the ACC controller implemented by this ADP algorithm is clearly demonstrated that experience replay can increase data efficiency significantly, and the approximate optimality and adaptability of the learned control policy are tested with typical driving scenarios.

Keywords Adaptive cruise control · Adaptive dynamic programming · Experience replay · Reinforcement learning · Neural networks

1 Introduction

After decades' investigation, advanced driver assistance systems (ADAS) have been built to improve driving safety, reduce fuel consumption and increase traffic capacity (Tapani 2012). The adaptive cruise control (ACC) system is developed from the traditional cruise control (CC) system, which constitutes a class of ADAS. For a following vehicle equipped with the ACC system, sensing information such as the relative speed and distance to a preceding vehicle is measured by sensors, and utilized to compute the corresponding throttle angle and brake pressure. Thus, the ACC strategy guides the vehicle to follow the preceding vehicle safely.

Considering the complexity and uncertainty of vehicle dynamics, the ACC problem is usually transformed to an optimal tracking control problem for complex nonlinear systems. Plenty of control strategies have been investigated to deal with the ACC problem, some of which are PID control (Guvenc and Kural 2006), model predictive control (Shakouri and Ordys 2014), sliding mode control (Xiao and Gao 2011), and fuzzy logic (Tsai et al. 2010). When designing the ACC controllers, the aforementioned works mainly require on a linear or nonlinear vehicle model, and the accuracy of the vehicle model determines the practical control performance of the ACC system. However, it is very hard to get an accurate vehicle model considering the complicated driving environment.

In this paper, we present a model-free reinforcement learning (RL) approach for the ACC problem. Reinforcement learning (RL) is an advanced learning control approach in dealing with complex decision-making problems. RL agent

Communicated by V. Loia.

✉ Bin Wang
cse_wangb@ujn.edu.cn

Dongbin Zhao
dongbin.zhao@ia.ac.cn

Jin Cheng
cse_chengj@ujn.edu.cn

[1] School of Electrical Engineering, University of Jinan, Jinan 250022, China

[2] The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Published online: 13 February 2018

interacts with the environment to get a reward and approximates an optimal policy by trial and error (Sutton and Barto 1998). Typical RL algorithms include Q-learning (Watkins and Dayan 1992), SARSA (Rummery and Niranjan 1994), actor-critic (Grondman et al. 2012), adaptive dynamic programming (ADP) (Liu et al. 2012; Zhao et al. 2015a), and so on. Reinforcement learning methods can learn the optimal control policy by interacting with the environment, without the knowledge of the system dynamics. RL approaches are successfully applied in robotics (Si and Wang 2001; Busoniu et al. 2010), intelligent transportation systems (Zhao et al. 2011), tracking problems (Kiumarsi and Lewis 2015), game problem (Zhao et al. 2012, 2015b), etc. Despite the adaptive learning capability, traditional RL approaches usually converge slowly for lacking of data efficiency, which is the main restriction for real-time applications.

Experience replay (ER) is viewed as a promising approach to improve RL efficiency. In traditional RL process, the experience obtained by trial and error is used to regulate parameters, but only used once and abandoned. Some of the experience is difficult to be reproduced in a real system, so it must be reused effectively, which is the concept of experience replay. Experience replay is first proposed in Lin (1992) under the background of neural network approximation based RL. The historical data such as state, action and reward are recorded, randomly chosen and repeatedly used in the value function estimation and policy update process. It is pointed out that actor-critic learning approach can be combined with experience replay, but not change the convergence property. A class of actor-critic algorithm with experience replay is summarized in Wawrzyński and Tanwani (2013). In Adam et al. (2012), ER-Q and ER-SARSA learning methods are proposed by integrating experience replay and Q-learning and SARSA, which perform well in a range of control problems, such as the inverted pendulum and a robot arm. A concurrent learning approach is proposed in Chowdhary and Johnson (2010) to solve the adaptive control problem of the uncertain system, in which the parameters update through the recorded historical data and current data, similar to experience replay. A event-triggered concurrent learning algorithm is proposed in Zhang et al. (2016), in which only one critic neural network is adopted to approximate both the value function and the policy. In recent years, as deep learning is becoming a central issue, experience replay is applied in deep learning. In Mnih et al. (2015), a deep Q-learning approach with experience replay is proposed to solve the high-dimensional Atari game problem.

In this paper, a model-free adaptive dynamic programming approach, combined with a new experience replay mechanism, is proposed to design an ACC controller. The contribution is threefold. First, a model-free ADP algorithm with experience replay (ADPER) is proposed to deal with the approximate optimal control problems for the unknown nonlinear systems. Second, the main feature of the experience replay mechanism is that when estimating parameters with gradient rules, the gradients of the recorded data and current data are used to update parameters concurrently. Both the critic part and the actor part of the ADP algorithm are integrated with ER. In addition, theoretical analysis is given by Lyapunov theory to prove that the estimation errors of the weights in the actor network and the critic network are uniformly ultimately bounded (UUB) under the novel weight update rules with ER. Third, based on this model-free algorithm, we design an ACC system to learn the acceleration controller, which improves the learning efficiency remarkably. Experimental results verify the effectiveness of the proposed algorithm under typical driving scenarios. The rest of the paper is as follows. We describe the ACC framework in Sect. 2. The implementation of the ADPER approach are presented in Sect. 3. In Sect. 4, the stability analysis of the algorithm is presented. Section 5 provides simulation results. We get to the conclusion in Sect. 6.

2 Adaptive cruise control problem

The ACC systems use radar and other sensory devices to detect the relative speed and distance to the preceding vehicle. The relative speed and distance are adjusted according to the behavior of the preceding vehicle, by properly modulating the throttle or applying the brake. The ACC controller is the core module of an ACC system, and intelligent control algorithms are of the research interest for decades. The capability of adapting to various driving scenarios and different drivers for the ACC controller is a main issue for the ACC system.

The control framework of the ACC system is shown in Fig. 1. Under the control of the ACC system, the following vehicle tries to keep a desired distance d_d and safe speed v_f with the preceding vehicle. The relative distance d_r acquired from sensors is used to compute the preceding vehicle speed v_p. With the instant information v_f, v_p, d_r and d_d, the ACC system generates appropriate control action to regulate the following vehicle. For the ACC problem, the control objective is to follow the driving behavior of the preceding vehicle with a desired clearance. As shown in Fig. 1, a hierarchical control structure is investigated to design the ACC system, which is a general design framework in this field (Moon et al. 2009). The upper level controller is our main concern in this paper. The lower level controller is introduced in detail in Wang et al. (2015).

For the acceleration controller in the upper level, the states are the relative speed Δv and the clearance Δd, and the output is the desired acceleration a_d. Thus, the tracking problem is transformed to a stabilization control problem. A state-space model for the controlled and preceding vehicles is described in Moon et al. (2009), and the ACC problem is viewed as

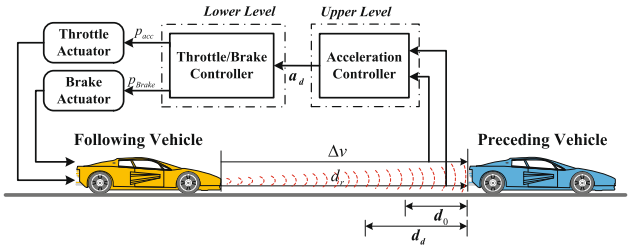

Fig. 1 The hierarchical control framework of the ACC system

a linear-quadratic optimization problem. When designing the desired acceleration controller, a state-feedback control principle is presented and the control gains are tuned under different driving modes, which seems like a piecewise adaptive control principle. Differently, to minimize the speed error Δv and the following-distance error Δd in the upper level controller in this paper, we will investigate a driving data-based approach to learn an adaptive acceleration principle in the full speed range without any vehicle model information.

The relative speed Δv is defined as

$$\Delta v = v_f - v_p \tag{1}$$

and the relative distance Δd is

$$\Delta d = d_r - d_d \tag{2}$$

where d_d is the desired clearance and can be described as

$$d_d = d_0 + v_p \cdot \tau \tag{3}$$

where τ is a constant time gap, which is an important index to measure driver character. A time gap varying from 1 to 2.3 seconds is used for many automobile manufacturers (Kyongsu and Ilki 2004). d_0 is the minimum braking distance.

It should be noted that the upper level controller reflects the driving behavior and preference of drivers, while conventional controllers cannot adapt to different driving habits. Although there are only two states in the upper level controller, driving behavior learning is still a key problem for an ACC system (Kyongsu and Ilki 2004; Moon et al. 2009). Therefore, considering the general adaptability of the controller, we propose a model-free adaptive dynamic programming approach with experience replay to design the upper level controller in the following. The design details of the lower level controller is presented in Wang et al. (2015).

3 The ADP approach with experience replay

In this section, we propose an ADP approach to learn the ACC controller, and a new experience replay technique is investigated to improve the learning efficiency. In Modares et al. (2014), an experience replay based actor-critic method is proposed in order to deal with partially unknown systems, in which experience replay only functions in the critic part. In this paper, differently, when estimating parameters with gradient rules for the actor part and the critic part in the ADP algorithm, we use both the gradients of historical data and current data. Moreover, feedforward neural networks with only one hidden layer are used to approximate the critic and the actor parts in this algorithm, respectively.

For the ACC problem, the actor part of the ADPER algorithm uses the current driving data (namely the states) to generate a desired acceleration and acts to the target vehicle. Then, the states transit to the next and an immediate reward is given. The critic part estimates the control policy and directs the policy improvement process. Details of the ADPER algorithm are as follows.

3.1 The critic network

The critic network evaluates the current control action by the future accumulative reward-to-go

$$R(t) = \sum_{k=0}^{T} \gamma^k r(t+k+1) \tag{4}$$

where $0 < \gamma \leq 1$ is a discount factor, t is the time step, $r(t)$ is the immediate reward, and T is the terminal step. Note that the reward value cannot be computed forward in time. Hence, a feedforward neural network with only a hidden layer is used to approximate $R(t)$ in (4), which is called value function as well.

The critic network outputs $\hat{Q}(t)$ to approximate $R(t)$ as follows

$$\hat{Q}(t) = \hat{w}_{\mathrm{c}}^T(t)\phi\left(v_{\mathrm{c}}^T X_c(t)\right) = \hat{w}_{\mathrm{c}}^T(t)\phi_{\mathrm{c}}(t) \tag{5}$$

where $X_{\mathrm{c}}(t) = [X_{\mathrm{a}}(t), u(t)]$ is the input of the critic network, $X_{\mathrm{a}}(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the action vector. For the ACC problem in the upper level, $X_{\mathrm{a}}(t) = [\Delta v(t), \Delta d(t)]$, $u(t) = a_d(t)$. $v_{\mathrm{c}} \in \mathbb{R}^{(n+m)\times s_1}$ is the ideal weight vector between the input layer and the hidden layer. $\hat{w}_{\mathrm{c}}(t) \in \mathbb{R}^{s_1 \times m}$ is the estimated weight vector between the hidden layer and the output layer. s_1 is the number of hidden layer neurons. Usually, the hidden layer weight vector v_{c} is randomly initialized and kept constant, so the activation function $\phi\left(v_{\mathrm{c}}^T X_c(t)\right)$ is rewritten as $\phi_{\mathrm{c}}(t)$ for short. In addition, a hyperbolic tangent function is used as the activation function in the networks.

The estimation error of the critic network weights $\tilde{w}_c(t)$ is defined as

$$\tilde{w}_c(t) = \hat{w}_c(t) - w_c \tag{6}$$

where $w_c \in \mathbb{R}^{s_2 \times m}$ is the ideal weight vector between the hidden layer and the output layer.

The prediction error of the critic network can be described as

$$e_c(t) = \gamma \hat{Q}(t) + r(t) - \hat{Q}(t-1) \tag{7}$$

Then, the objective function to be minimized by the critic network is defined as

$$E_c(t) = \frac{1}{2} e_c^2(t) \tag{8}$$

When updating the weights of the critic network with a gradient rule, a novel weight update law is proposed to take advantage of experience replay. The gradient of the weight vector is defined as follows

$$\begin{aligned}\Delta \hat{w}_c(t) =& -\beta\gamma\phi_c(t)e_c^T(t) - \beta\gamma\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t) \\ =& -\beta\gamma\phi_c(t)\Big[\gamma\hat{w}_c^T(t)\phi_c(t) + r(t) \\ & -\hat{w}_c^T(t-1)\phi_c(t-1)\Big]^T \\ & -\beta\gamma\sum_{j=1}^{l}\phi_{c_j}(t)\Big[\gamma\hat{w}_c^T(t)\phi_{c_j}(t) \\ & + r_j(t) - \hat{w}_c^T(t-1)\phi_{c_j}(t-1)\Big]^T \end{aligned} \tag{9}$$

where $\beta > 0$ is the learning rate of the critic network.

Note that in (9), t is the current step, and j is the jth sample data stored in database, where $j = 1, \ldots, l$, and l is the length of the database.

Correspondingly, the weight update of the critic network is given as

$$\hat{w}_c(t+1) = \hat{w}_c(t) + \Delta\hat{w}_c(t) \tag{10}$$

3.2 The actor network

In this section, an actor network is used to approximate the optimal control policy without any modeling information of vehicle dynamics. Actor network is constructed by a feed-forward neural network with one hidden layer, and inputs of the actor network are the error states of the vehicle, while output is the control action, namely a desired acceleration.

Output of the actor network can be described as

$$u(t) = \hat{w}_a^T(t)\phi\left(v_a^T X_a(t)\right) = \hat{w}_a^T(t)\phi_a(t) \tag{11}$$

where $X_a(t) = [\Delta v(t), \Delta d(t)]$ is the input vector of the actor network, $v_a \in \mathbb{R}^{n \times s_2}$ is the ideal weight vector between the input layer and the hidden layer. $\hat{w}_a(t) \in \mathbb{R}^{s_2 \times m}$ is the estimated weight vector between the hidden layer and the output layer. s_2 is the number of hidden layer neurons. Since the hidden layer weight vector v_a is randomly initialized and kept constant, the activation function $\phi\left(v_a^T X_a(t)\right)$ is rewritten as $\phi_a(t)$ for short.

The estimation error of the actor network weights $\tilde{w}_a(t)$ is defined as

$$\tilde{w}_a(t) = \hat{w}_a(t) - w_a \tag{12}$$

where $w_a \in \mathbb{R}^{s_1 \times m}$ is the ideal weight vector between the hidden layer and the output layer.

The prediction error of the actor network is proposed by

$$e_a(t) = \hat{Q}(t) \tag{13}$$

The objective function to be minimized by the actor network is defined as

$$E_a(t) = \frac{1}{2} e_a^2(t) \tag{14}$$

Similarly, the weight update law for the actor network is given by

$$\begin{aligned}\Delta\hat{w}_a(t) =& -\alpha\phi_a(t)C(t)\Big[\hat{w}_c^T(t)\phi_c(t)\Big]^T \\ & -\sum_{j=1}^{l}\alpha\phi_{a_j}(t)C_j(t)\Big[\hat{w}_c^T(t)\phi_{c_j}(t)\Big]^T \end{aligned} \tag{15}$$

where $\alpha > 0$ is the learning rate of the actor network, $C(t) \in \mathbb{R}^{m \times s_2}$ is weight vector of the critic network connected to the control action $u(t)$. In (15), t is the current step, while j is the jth sample data stored in database, where $j = 1, \ldots, l$, and l is the length of the database.

Then, weight update of the actor network is as follows

$$\hat{w}_a(t+1) = \hat{w}_a(t) + \Delta\hat{w}_a(t) \tag{16}$$

The learning procedure of the ADPER algorithm proposed in this section is as Algorithm 1. First, initialize the learning parameters of the neural networks. During the ADP learning process, store the sample data into the database and make sure only l most recent data remain in the database. Then when update the parameters of the neural networks, compute the current gradients and compute the historical gradients in the database concurrently. At last, sum of the current gradients and the historical gradients is used to update the weights of the actor network and the critic network. Different from Modares et al. (2014), in which experience replay only functions in critic network, experience replay in our proposed

ADPER algorithm acts both in critic network and in actor network. In next section, the stability property of the ADPER algorithm is given in detail.

Algorithm 1 ADP with experience replay

Initialization
actor network weights v_a, w_a, learning rate α
critic network weights v_c, w_c, learning rateβ
discount factor $\gamma \in [0, 1]$, database $D \leftarrow \emptyset$, length l
$x(0) \leftarrow$initial state
for each step k **do**
$u(t) \leftarrow$actor network output
take action $u(t)$, **observe** reward $r(t+1)$, and the next state $x(k+1)$
add the data sample to the database $D \leftarrow D \cup \{\langle x(t), u(t), x(t+1), r(t+1)\rangle\}$
make sure only l most recent data stored in D
compute the current gradients of the critic network weights by (9)
compute the current gradients of the actor network weights by (15)
loop l times
choose a tuple (x, u, x', r) randomly in D
compute the gradients of the critic network weights by (9)
compute the gradients of the actor network weights by (15)
end loop
update the weights of the critic network by (10)
update the weights of the actor network by (16)
end for

4 Stability analysis of the ADPER algorithm

In this section, we analyze the stability of the ADPER algorithm with Lyapunov method. Under the framework delineated in Dierks et al. (2009); Liu et al. (2012); Zhao et al. (2013); Yang et al. (2014); Kang et al. (2016, 2015), we prove that the weight estimation errors of the actor network and the critic network are still uniformly ultimately bounded (UUB) under the novel weight update rules proposed in this paper. Before demonstrating the main theorem, the following assumption and fact are given.

Assumption 1 *The ideal weights w_a and w_c in the actor network and the critic network are bounded over the compact set $\Omega \in \mathbb{R}$, namely there exist constants $w_{am} > 0$ and $w_{cm} > 0$ such that*

$$\|w_a\| \le w_{am}, \quad \|w_c\| \le w_{cm} \tag{17}$$

Fact 1 *The activation functions $\phi_a(t)$ and $\phi_c(t)$ for the actor network and the critic network are bounded over the compact set $\Omega \in \mathbb{R}$. That is, there exist $\phi_{am} > 0$ and $\phi_{cm} > 0$ such that*

$$\|\phi_a\| \le \phi_{am}, \quad \|\phi_c\| \le \phi_{cm} \tag{18}$$

Then, several lemmas are developed by using Assumption 1 and Fact 1, which are the bases to derive our main theorem.

Lemma 1 *Let Assumption 1 holds. Taking the output of the critic network as (5), and the weights update rule is as (9) and (10), then for*

$$L_1(t) = \frac{\lambda_1}{\beta} tr\left(\tilde{w}_c^T(t)\tilde{w}_c(t)\right) \tag{19}$$

its first difference satisfies

$$\begin{aligned}\Delta L_1(t) \le & -\lambda_1\gamma^2\left(1-\beta\gamma^2\|\phi_c(t)\|^2\right)\|\xi_c(t)\|^2 \\ & -\lambda_1\left(1-\beta\gamma^2\|\phi_c(t)\|^2\right)\left\|\gamma\xi_c^T(t)+\gamma w_c^T(t)\phi_c(t)\right. \\ & \left.+\, r(t)-\hat{w}_c^T(t-1)\phi_c(t-1)\right\|^2 \\ & +\lambda_1\left(1+\beta\gamma^2\|\phi_c(t)\|^2\right)\left\|\gamma w_c^T(t)\phi_c(t)+r(t)\right. \\ & \left.-\,\hat{w}_c^T(t-1)\phi_c(t-1)\right\|^2+\lambda_1\left(1+3\beta\gamma^2\right) \\ & \times\left\|\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2-\lambda_1\beta\gamma^2\left\|\gamma\phi_c(t)\xi_c^T(t)\right. \\ & \left.-\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2+\lambda_1\gamma^2\|\tilde{w}_c(t)\|^2 \end{aligned} \tag{20}$$

where $\lambda_1 > 0$ is regulating constant, and $\xi_c(t) = \tilde{w}_c^T(t)\phi_c(t) = \left(\hat{w}_c(t) - w_c\right)^T\phi_c(t)$

Proof See in Appendix A. □

Lemma 2 *Let Assumption 1 holds. Taking the output of the actor network as (11), and the weight update rule is as (15) and (16), then for*

$$L_2(t) = \frac{\lambda_2\gamma^2}{\alpha} tr\left(\tilde{w}_a^T(t)\tilde{w}_a(t)\right) \tag{21}$$

its first difference satisfies

$$\begin{aligned}\Delta L_2(t) \le & -\lambda_2\gamma^2\|C(t)\|^2\|\xi_a(t)\|^2-2\lambda_2\gamma^2 \\ & \times\left(1-\alpha\|C(t)\|^2\|\phi_a(t)\|^2\right)\left\|\hat{w}_c^T(t)\phi_c(t)\right\|^2 \\ & -\lambda_2\gamma^2\left\|\tilde{w}_a^T(t)+\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2 \\ & +2\lambda_2\gamma^2\|\xi_c(t)\|^2+2\lambda_2\gamma^2\|w_c^T\phi_c(t)\|^2+\lambda_2\gamma^2 \\ & \times\|\tilde{w}_a(t)\|^2+\lambda_2\gamma^2\left\|\xi_a(t)C(t)-\hat{w}_c^T(t)\phi_c(t)\right\|^2 \end{aligned}$$

$$+ \lambda_2 \gamma^2 (1+2\alpha) \left\| \sum_{j=1}^{l} \phi_{a_j}(t) C_j(t) e_{a_j}^T(t) \right\|^2 \quad (22)$$

where $\lambda_2 > 0$ *is a regulating constant, and* $\xi_{\mathrm{a}}(t) = \tilde{w}_{\mathrm{a}}^T(t)\phi_{\mathrm{a}}(t) = \left(\hat{w}_{\mathrm{a}}(t) - w_{\mathrm{a}}\right)^T \phi_{\mathrm{a}}(t)$.

Proof See in Appendix B. □

Next combining Assumption 1, Fact 1, Lemmas 1 and 2, we propose the main theorem of this paper.

Theorem 1 *Let Assumption 1 hold. The weight update rules for the critic network* (5) *and the actor network* (11) *are as* (9), (10), *and* (15), (16), *respectively. Then, provided the following conditions hold*

$$(a) \quad 0 < \alpha \|C(t)\|^2 \|\phi_{\mathrm{a}}(t)\|^2 < 1$$
$$(b) \quad 0 < \beta \gamma^2 \|\phi_{\mathrm{c}}(t)\|^2 < 1$$

the weight estimation errors of the critic network $\tilde{w}_{\mathrm{c}}(t)$ *and the actor network* $\tilde{w}_{\mathrm{a}}(t)$ *are UUB by positive constants* $\mathfrak{B}_{\mathrm{c}}$ *and* $\mathfrak{B}_{\mathrm{a}}$, *respectively, which are given by*

$$\mathfrak{B}_{\mathrm{c}} = \frac{1}{\phi_{\mathrm{cm}}} \sqrt{\frac{\mathfrak{L}_m^2}{\gamma^2 \left[\lambda_1 \left(1 - \beta \gamma^2 \|\phi_{\mathrm{c}}(t)\|^2\right) - 2\lambda_2\right]}}$$
$$\mathfrak{B}_{\mathrm{a}} = \frac{1}{\phi_{\mathrm{am}}} \sqrt{\frac{\mathfrak{L}_m^2}{\lambda_2 \gamma^2 \|C(t)\|^2}}$$

where $\lambda_i > 0$ $(i = 1, 2)$ *are regulating constants.*

Proof See in "Appendix C". □

5 Experimental results

In this section, we design the ACC controller with the ADPER algorithm to verify the effectiveness of experience replay for improving the learning efficiency of ADP. Then, the learned controller is tested in dSPACE simulator for the adaptability of various driving scenarios.

5.1 dSPACE simulator

dSPACE simulator builds a real-time simulation and development platform for car makers. Complete automotive simulation models (ASM) are provided in dSPACE for real-time simulation of vehicle dynamics (dSPACE 2015), which include an accurate physical vehicle model, driving environments with road, maneuvers and drivers. dSPACE simulator enables the ACC system to be evaluated in a virtual environment, which can avert risk for the host and other vehicles under real traffic conditions.

We design the ACC controller in the Soft ECU block of dSPACE. Moreover, in the Environment block, we construct roads and driving maneuvers to evaluate performance of the proposed ACC system.

5.2 The design of ACC learning with ADPER

The main implementation framework of the ACC system is described in Wang et al. (2015). With similar configuration, the ADPER algorithm is developed to learn the upper level controller of the ACC system. Different from Zhao et al. (2013), which adopts a simple vehicle dynamics model to provide training data, the dSPACE simulator is used to design vivid driving scenario for learning.

When learning the ACC strategy with ADPER algorithm, we define the following terminal state

$$\begin{cases} |\Delta v| < 0.072 km/h \\ |\Delta d| < 0.2m \end{cases} \quad (23)$$

The reward value is defined as

$$r(t) = \begin{cases} 0, & terminal\ state \\ -1, & other\ state \\ -10, & d_r \le 0 \end{cases} \quad (24)$$

The reward signal is assigned "0" for "success" when terminal state occurs, and " − 1" for other states. A bump state is defined as $d_r \le 0$, and the reward value " − 10" is given for punishment which means "failure" under this state.

With the ADPER learning algorithm, a flow chart of the ACC system is shown in Fig. 2. During driving, once the ACC system starts the sensors equipped in the vehicle detects whether there is a preceding target vehicle. If not, the vehicle cruises with a constant set speed, by an appropriate throttle torque to the throttle actuator. When a target vehicle is detected, the ADPER algorithm in the upper level learns a control action by the sensor data and a desired acceleration is output to the lower level controller. The throttle and/or brake controller in the lower level computes the required throttle or brake torque, which is acted directly on the vehicle by the engine or brake actuator.

5.3 Learning performance of the ADPER algorithm

Training of the ADPER algorithm follows the procedure of Algorithm 1. The discount factor γ is 0.95, the learning rate α is 0.001, and β is 0.001. The number of hidden layer neurons is configured 8 experientially. Weights of the neural networks are all initialized randomly. The length of the database for experience replay is configured 10 experimentally.

After each iteration during the learning process, the performance of the current policy is evaluated and recorded.

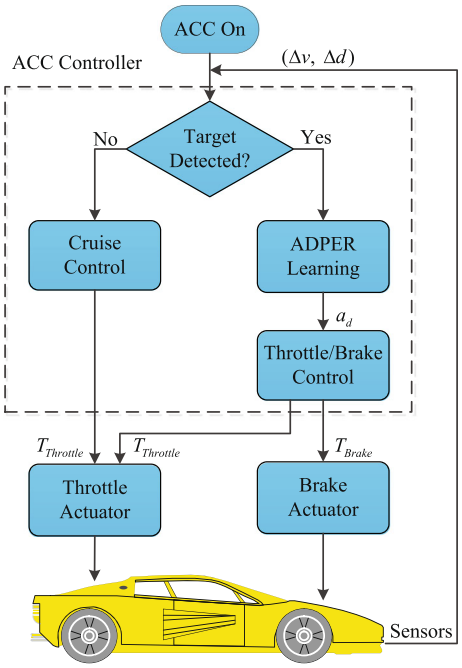

Fig. 2 The ACC system control flow chart with ADPER algorithm

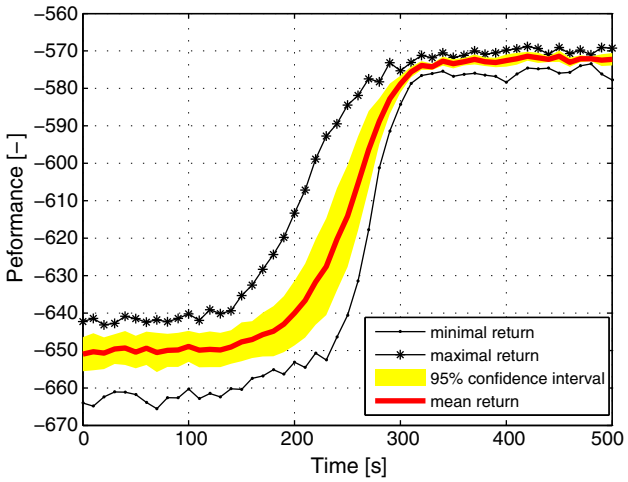

Fig. 3 The learning performance of the ADPER algorithm

The learning and exploration process is suspended when performance evaluation. And the performance of the policy is obtained by testing a group of initial states to calculate the average return. To verify the efficiency of the ADPER algorithm, a typical ADP algorithm is investigated with the same configuration. 10 experiments are conducted for both of the algorithms and the mean returns are acquired respectively, which are shown in Figs. 3 and 4.

The learning performance of the ADPER algorithm is shown in Fig. 3, including minimal return, mean return, maximal return and 95% confidence interval of the mean return. Note that the converged return is acquired at about 400 s, which means a nearly optimal policy is learned. Compared

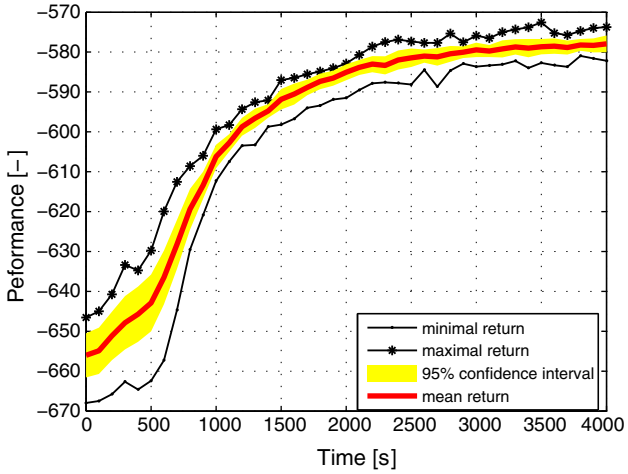

Fig. 4 The learning performance of typical ADP algorithm

with the performance of classical ADP algorithm shown in Fig. 4, which needs about 3000 s to get the converged return. We can draw the conclusion that experience replay can improve the learning efficiency significantly and make the ADP algorithm converge faster. Furthermore, the converged return means a nearly optimal policy, which indicates the weights of the neural networks are converged and bounded.

5.4 Adaptability test under different driving scenarios

To verify the performance of the control policy getting from the ADPER algorithm, the actor network is used as acceleration controller after training to test the performance under typical driving scenarios, including normal driving, cut-in/cut-out, emergency braking and so on. For comparison purpose, a well-tuned PID controller and another neural network controller learned by supervised ADP (SADP) algorithm in Zhao et al. (2013) are investigated in this section. And main comparison indexes include vehicle speed, clearance, acceleration, inverse TTC (time-to-collision) (Moon et al. 2009), throttle angle, and brake pressure. Experimental results provided by dSPACE simulator are shown as follows. Note that to highlight the performance of the proposed ACC system, it is set to start the ACC operation when $t = 20$ s.

5.4.1 Natural driving scenario

Figure 5 shows the comparison results of the natural driving scenario. The preceding vehicle starts and accelerates to 72 km/h. At $t = 63$ s the preceding vehicle accelerates to 90 km/h in 10 s, and holds about 40 s. Then, it decelerates to 54 km/h at $t = 113$ s. The following vehicle controlled by the ACC system starts at $100m$ away from the preceding vehicle at the same time with initial set speed 100 km/h. When

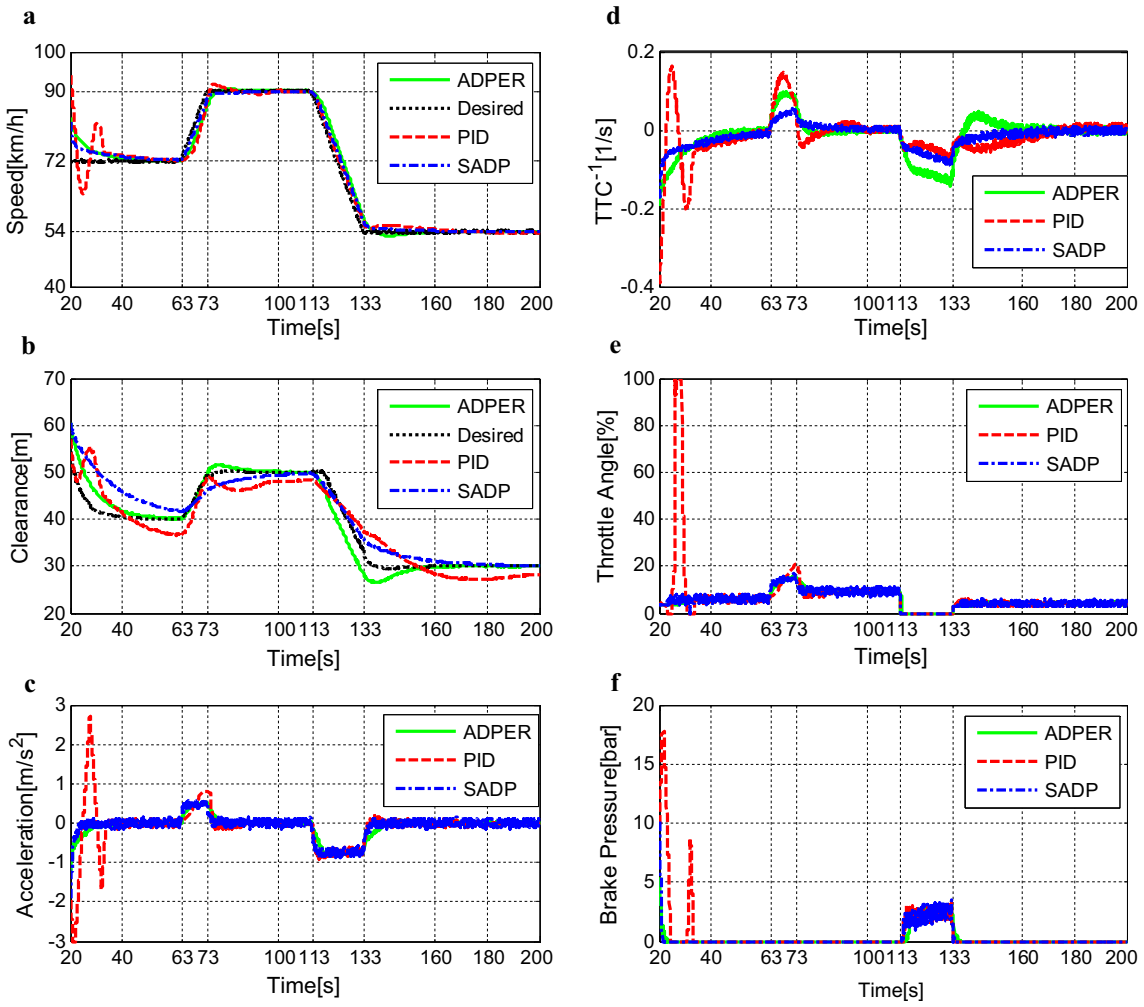

Fig. 5 Experimental results in the natural driving scenario. **a** Vehicle speed, **b** clearance, **c** acceleration, **d** TTC^{-1}, **e** throttle angle, and **f** brake pressure

$t = 20$ s the ACC system starts, and after the preceding vehicle is detected by sensors, the following vehicle tracks the preceding vehicle to achieve desired speed and safety distance under the ACC system, as is shown in Fig. 5a, b. The desired accelerations output by the upper level controller are compared in Fig. 5c.

It can be concluded that for the following vehicle the ADPER controller performs as well as PID control and SADP algorithm in speed, but better in clearance maintenances with smaller steady state error. Compared with SADP algorithm, there is a little overshoot in distance control for the ADPER approach but acceptable. The inverse TTC shown in Fig. 5d indicates that the index changes in a small domain for all three controllers, which means the current driving situation is safe. Moreover, as shown in Fig. 5e, f, when the preceding vehicle decelerates at $t = 113$ s, the throttle angle is set to zero and the brake is applied to reduce the speed.

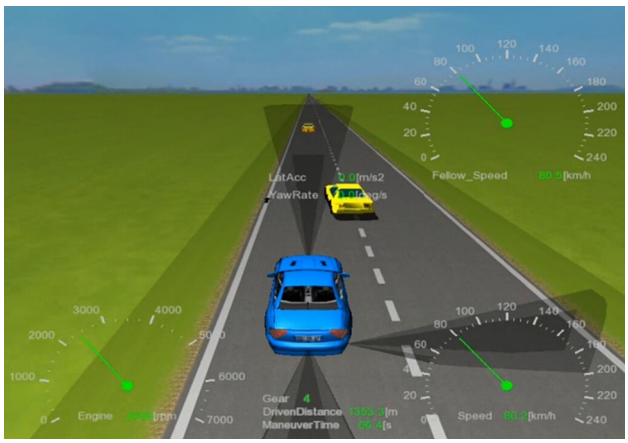

Fig. 6 The motion display of the cut-in and cut-out scenarios

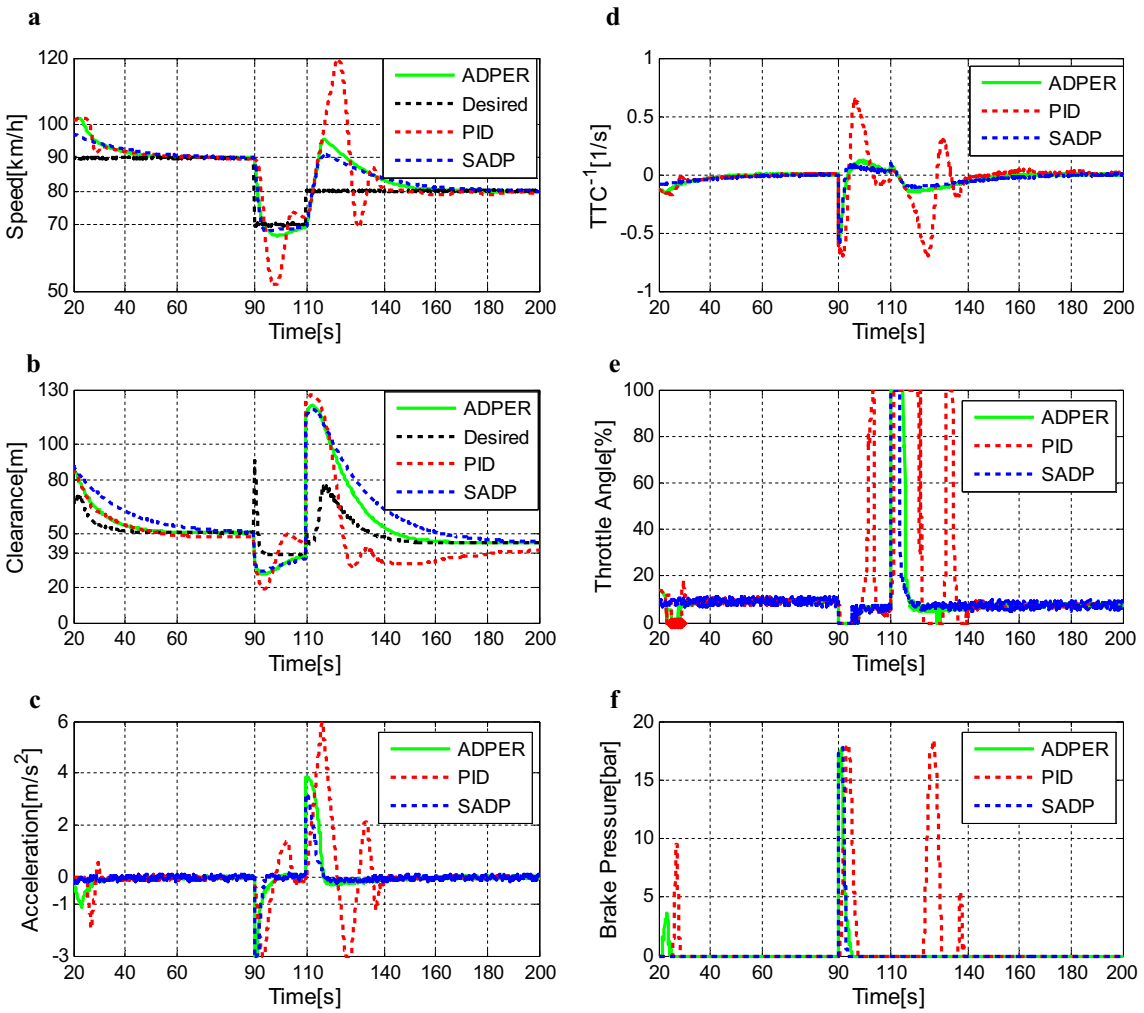

Fig. 7 Experimental results in the cut-in and cut-out scenarios. **a** Vehicle speed, **b** clearance, **c** acceleration, **d** TTC^{-1}, **e** throttle angle, and **f** brake pressure

5.4.2 Cut-in and cut-out scenarios

Diagrammatic description of the cut-in and cut-out scenarios is shown in Fig. 6. In Fig. 7, the control performance in cut-in and cut-out scenarios is presented. The preceding vehicle travels by 90 km/h, while the following vehicle follows with about 100 km/h. At $t = 90$ s another vehicle cuts in with 70 km/h. At this time, the inserted vehicle is detected by sensors as a new preceding vehicle, and there is a sudden change for both the speed and the clearance. Therefore, the following vehicle decelerates to track the behavior of the inserted vehicle. Then, at $t = 110$ s, the inserted vehicle cuts out while the preceding vehicle travels at 80 km/h. Thus, the following vehicle accelerates to keep up with the preceding vehicle again. The vehicle speeds and clearances are compared in Fig. 7a, b, respectively. We comment that in this scenario, the ADPER algorithm performs as well as SADP method.

Compared with PID controller, the ADPER algorithm performs better in regulating both the speed and the distance. The distance error from the desired safety distance is more than 6 m for the PID controller, which increases the risk of crash. As shown in Fig. 7c, the desired acceleration oscillates in a wider range for the PID-based ACC system, which affects the driving comfort of the drivers. When the cut-in vehicle appears at $t = 90$ s, the throttle angle is set to zero and the brake is applied to decelerate the vehicle to avoid collision. While when the vehicle cuts out at $t = 110$ s, the brake is set to zero and the throttle angle increases to accelerate the vehicle to decrease the clearance, as shown in Fig. 7e, f. Figure 7d shows that the inverse TTC decreases abruptly when the cut-in vehicle appears.

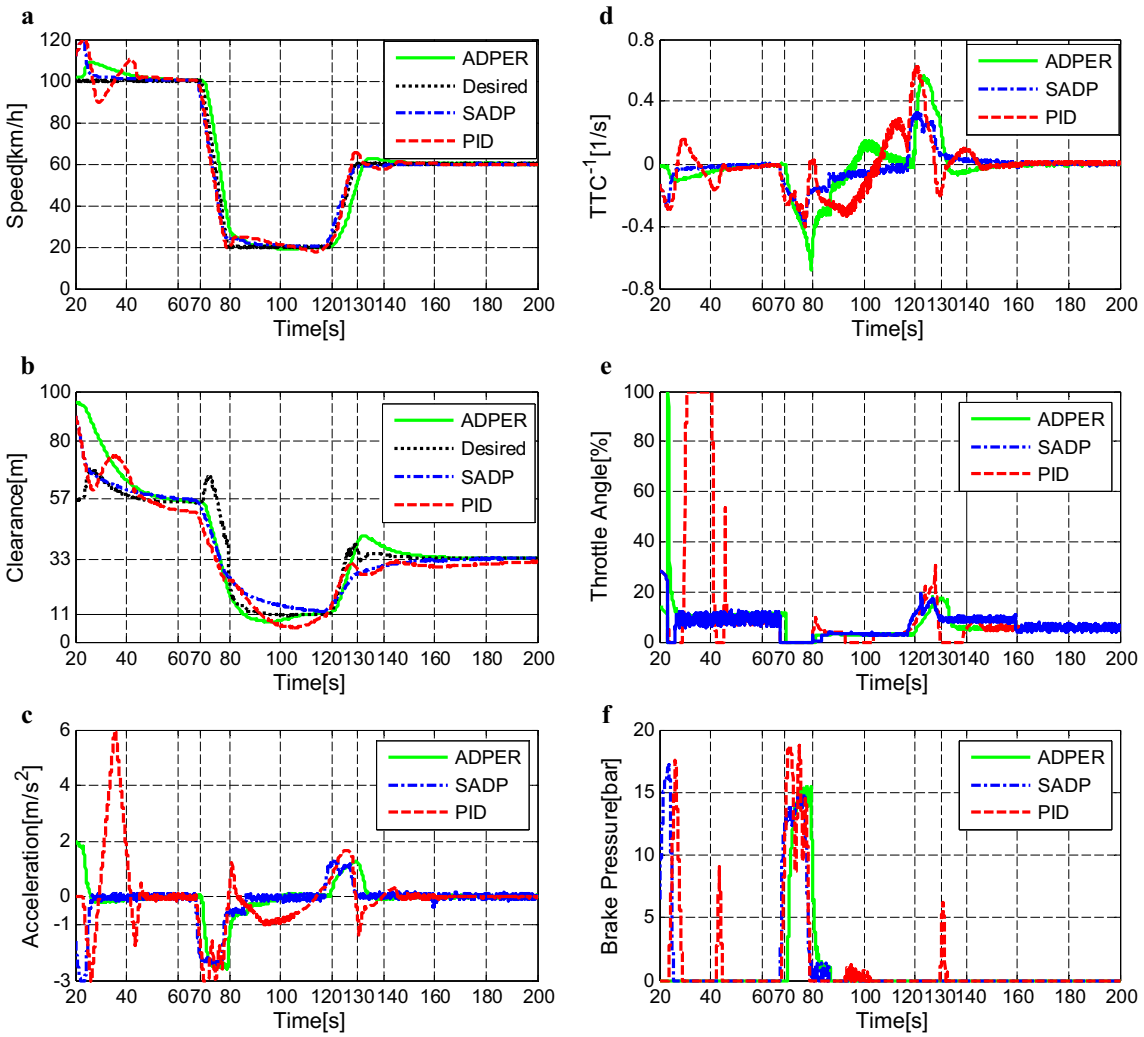

Fig. 8 Experimental results in the emergency braking scenario. **a** Vehicle speed, **b** clearance, **c** acceleration, **d** TTC^{-1}, **e** throttle angle, and **f** brake pressure

5.4.3 Emergency braking scenario

The control performance of the emergency braking scenario is presented in Fig. 8. In Fig. 8a, the preceding vehicle travels with 100 km/h and decelerates to 20 km/h at $t = 70$ s. Then, the preceding vehicle accelerates to 60 km/h at $t = 120$ s. The following vehicle brakes as soon as this change is detected. The relative distance is shown in Fig. 8b, which describes the driving safety. We conclude that the ADPER approach maintains a more safe clearance than the PID controller, which improves the safety for drivers during braking. Although there is some overshoot for the ADPER algorithm, the overall control performance is as good as SADP approach.

The change of the acceleration during the emergency braking process is an important index to measure driving comfort. Figure 8c indicates that the acceleration of the proposed the ACC system with the ADPER algorithm changes more gently than PID controller, which improves the driving comfort significantly. In Fig. 8e, f, the throttle angle is set to zero and the brake is applied to avoid collision with the preceding vehicle, but the brake pressure output by the ADPER-based ACC system is milder. Figure 8d implies that the inverse TTC abruptly decreases, and the driving safety is reduced. The ADPER approach can stabilize this index around 0 quickly; however, PID controller oscillates severely and affects driving safety obviously.

5.5 Discussions

The learning efficiency and remarkable performance of the proposed ADPER algorithm has been demonstrated in the above simulation experiments. During the learning process of the ACC controller, the ADPER approach selects and stores historical data online, and repeatedly uses them to conduct the parameter update procedure, which can improve the data efficiency and accelerate the learning speed of the algorithm

remarkably. Moreover, we conclude from the experimental results that compared with SADP algorithm, which is an advanced ACC algorithm as well, the ADPER algorithm does not bother the approximate optimality of the control policy.

6 Conclusions

In this paper, an efficient model-free ADP algorithm is developed to design the ACC controller. We have investigated the practical performance of a class of ADP methods combining with experience replay. The main contribution is that when estimating parameters of the critic network and the actor network with gradient rules for the ADP algorithm, the gradients of historical data and current data are used to update parameters concurrently, and this is the development of experience replay. It is proved with Lyapunov theory that the weight estimation errors of the actor network and the critic network are UUB. The ADPER algorithm is implemented with neural networks to design the ACC controller. The learning results indicate that experience replay can increase data efficiency of the ADP algorithm significantly, and most importantly, the approximate optimality and adaptability of the learned control policy are still guaranteed. In addition, the effectiveness of the developed algorithm is tested with typical driving scenarios.

For future work, considering the uncertainties of the real driving environment, a practical ACC system is to be developed to test the control policy on field. Hardware-in-the-loop or human-in-the-loop is a very important feature to implement an intelligent ACC system. Moreover, other efficient RL methods are our research interest.

Acknowledgements This work was supported partly by National Natural Science Foundation of China (Nos. 61603150, 61273136, 61573353 and 61533017), the National Key Research and Development Plan (No. 2016YFB0101000), and Doctoral Foundation of University of Jinan (No. XBS1605).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

Appendices

A Proof of Lemma 1

The first difference of $L_1(t)$ is

$$\Delta L_1(t) = \frac{\lambda_1}{\beta} tr\left[\tilde{w}_c^T(t+1)\tilde{w}_c(t+1) - \tilde{w}_c^T(t)\tilde{w}_c(t)\right]$$

$$= \frac{\lambda_1}{\beta} tr\left[(\tilde{w}_c(t) + \Delta\hat{w}_c(t))^T(\tilde{w}_c(t) + \Delta\hat{w}_c(t)) - \tilde{w}_c^T(t)\tilde{w}_c(t)\right] \quad (25)$$

with the basic property of matrix $tr(AB) = tr(BA)$, then

$$\Delta L_1(t) = \frac{\lambda_1}{\beta} tr\left[2\tilde{w}_c^T(t)\Delta\hat{w}_c(t) + \Delta\hat{w}_c^T(t)\Delta\hat{w}_c(t)\right] \quad (26)$$

Substituting (9) into (26), we obtain

$$\begin{aligned}\Delta L_1(t) =& \lambda_1 tr\Bigg[- 2\gamma\tilde{w}_c^T(t)\phi_c(t)e_c^T(t) \\ & - 2\gamma\tilde{w}_c^T(t)\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t) \\ & + \beta\gamma^2\left(\phi_c(t)e_c^T(t) + \sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right)^T \\ & \times\left(\phi_c(t)e_c^T(t) + \sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right)\Bigg] \\ =& \lambda_1 tr\left[\Re_1(t) + \Re_2(t)\right]\end{aligned} \quad (27)$$

where

$$\Re_1(t) = - 2\gamma\tilde{w}_c^T(t)\phi_c(t)e_c^T(t) - 2\gamma\tilde{w}_c^T(t)\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)$$

$$\Re_2(t) = \beta\gamma^2\left(\phi_c(t)e_c^T(t) + \sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right)^T \times\left(\phi_c(t)e_c^T(t) + \sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right)$$

with (7) we have

$$\begin{aligned}e_c(t) =& \gamma\hat{w}_c^T(t)\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1) \\ =& \gamma\xi_c(t) + \gamma w_c^T(t)\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1)\end{aligned} \quad (28)$$

For simplicity, we remark

$$P(t) = \gamma w_c^T\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1) \quad (29)$$

then

$$\Re_1(t) = - 2\gamma\xi_c(t)\left(\gamma\xi_c(t) + P(t)\right)^T - 2\gamma\tilde{w}_c^T(t)\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)$$

$$\begin{aligned}\mathfrak{R}_2(t) =& \beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\left(\gamma\xi_{\mathrm{c}}(t)+P(t)\right)\left(\gamma\xi_{\mathrm{c}}(t)+P(t)\right)^T\\ &+2\beta\gamma^2\phi_{\mathrm{c}}(t)(\gamma\xi_{\mathrm{c}}(t)+P(t))^T\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\\ &+\beta\gamma^2\left\|\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2\end{aligned}$$

Substituting $\mathfrak{R}_1(t)$ and $\mathfrak{R}_2(t)$ into (27), we obtain

$$\begin{aligned}\Delta L_1(t) =& -\lambda_1\gamma^2\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)\|\xi_{\mathrm{c}}(t)\|^2\\ &-\lambda_1\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)\|\gamma\xi_{\mathrm{c}}(t)+P(t)\|^2\\ &-\lambda_1\gamma^2\|\tilde{w}_{\mathrm{c}}(t)\|^2-\lambda_1\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)\\ &\times\left\|\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2-\lambda_1\beta\gamma^2\left\|\gamma\xi_{\mathrm{c}}(t)\phi_{\mathrm{c}}^T(t)\right.\\ &\left.-\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2+\lambda_1\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)\\ &\times\|P(t)\|^2+\lambda_1\left\|\gamma\tilde{w}_{\mathrm{c}}^T(t)-\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2\\ &+\lambda\beta\gamma^2\left\|P(t)\phi_{\mathrm{c}}^T(t)-\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2 \qquad (30)\end{aligned}$$

Applying the Cauchy–Schwarz inequality $\|x+y\|^2 \le 2\|x\|^2+2\|y\|^2$, we can derive (20).

The proof is completed.

B Proof of Lemma 2

The first difference of $L_2(t)$ is

$$\begin{aligned}\Delta L_2(t) =& \frac{\lambda_2\gamma^2}{\alpha}tr\left[\tilde{w}_{\mathrm{a}}^T(t+1)\tilde{w}_{\mathrm{a}}(t+1)-\tilde{w}_{\mathrm{a}}^T(t)\tilde{w}_{\mathrm{a}}(t)\right]\\ =& \frac{\lambda_2\gamma^2}{\alpha}tr\left[(\tilde{w}_{\mathrm{a}}(t)+\Delta\hat{w}_{\mathrm{a}}(t))^T(\tilde{w}_{\mathrm{a}}(t)+\Delta\hat{w}_{\mathrm{a}}(t))\right.\\ &\left.-\tilde{w}_{\mathrm{a}}^T(t)\tilde{w}_{\mathrm{a}}(t)\right]\\ =& \frac{\lambda_2\gamma^2}{\alpha}tr\left[2\tilde{w}_{\mathrm{a}}^T(t)\Delta\hat{w}_{\mathrm{a}}(t)+\Delta\hat{w}_{\mathrm{a}}^T(t)\Delta\hat{w}_{\mathrm{a}}(t)\right]\end{aligned} \qquad (31)$$

Substituting (15) into (31), we have

$$\begin{aligned}\Delta L_2(t) =& \lambda_2\gamma^2\left[\left\|\xi_{\mathrm{a}}(t)C(t)-\hat{w}_{\mathrm{c}}^T(t)\phi_{\mathrm{c}}(t)\right\|^2\right.\\ &-\|\hat{w}_{\mathrm{c}}^T(t)\phi_{\mathrm{c}}(t)\|^2-\|C(t)\|^2\|\xi_{\mathrm{a}}(t)\|^2+\|\tilde{w}_{\mathrm{a}}(t)\|^2\\ &-\left\|\tilde{w}_{\mathrm{a}}^T(t)+\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2\\ &+\left\|\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2\\ &\left.+\alpha\left\|\phi_{\mathrm{a}}(t)C(t)e_{\mathrm{a}}^T(t)+\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2\right]\end{aligned} \qquad (32)$$

By Cauchy–Schwarz inequality, we can derive (22).

The proof is completed.

C Proof of Theorem 1

Consider the following Lyapunov function candidate

$$L(t) = L_1(t)+L_2(t)$$

where

$$\begin{aligned}L_1(t) =& \frac{\lambda_1}{\beta}tr[\tilde{w}_{\mathrm{c}}^T(t)\tilde{w}_{\mathrm{c}}(t)]\\ L_2(t) =& \frac{\lambda_2}{\alpha}tr[\tilde{w}_{\mathrm{a}}^T(t)\tilde{w}_{\mathrm{a}}(t)]\end{aligned}$$

The first difference of $L(t)$ is

$$\Delta L(t) = L(t+1)-L(t) = \Delta L_1(t)+\Delta L_2(t) \qquad (33)$$

By employing Lemma 1 and Lemma 2, we obtain

$$\begin{aligned}\Delta L(t) \le& -\gamma^2\left[\lambda_1\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)-2\lambda_2\right]\|\xi_{\mathrm{c}}(t)\|^2\\ &-\lambda_1\left(1-\beta\gamma^2\|\phi_{\mathrm{c}}(t)\|^2\right)\left\|\gamma\xi_{\mathrm{c}}^T(t)+\gamma w_{\mathrm{c}}^T(t)\phi_{\mathrm{c}}(t)\right.\\ &\left.+r(t)-\hat{w}_{\mathrm{c}}^T(t-1)\phi_{\mathrm{c}}(t-1)\right\|^2\\ &-\lambda_1\beta\gamma^2\left\|\gamma\phi_{\mathrm{c}}(t)\xi_{\mathrm{c}}^T(t)-\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2\\ &-\lambda_2\gamma^2\|C(t)\|^2\|\xi_{\mathrm{a}}(t)\|^2\\ &-\lambda_2\gamma^2\left\|\tilde{w}_{\mathrm{a}}^T(t)+\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2\\ &-2\lambda_2\gamma^2\left(1-\alpha\|C(t)\|^2\|\phi_{\mathrm{a}}(t)\|^2\right)\left\|\hat{w}_{\mathrm{c}}^T(t)\phi_{\mathrm{c}}(t)\right\|^2\\ &+\mathfrak{L}^2(t)\end{aligned} \qquad (34)$$

where

$$\begin{aligned}\mathfrak{L}^2(t) =& \lambda_1\left(1+\beta\gamma^2\|\phi_{\rm c}(t)\|^2\right)\left\|\gamma w_{\rm c}^T(t)\phi_{\rm c}(t)+r(t)\right.\\ &\left.-\hat{w}_{\rm c}^T(t-1)\phi_{\rm c}(t-1)\right\|^2\\ &+\lambda_1\left(1+3\beta\gamma^2\right)\left\|\sum_{j=1}^{l}\phi_{c_j}(t)e_{c_j}^T(t)\right\|^2\\ &+\lambda_1\gamma^2\|\tilde{w}_{\rm c}(t)\|^2+\lambda_2\gamma^2\|\tilde{w}_{\rm a}(t)\|^2\\ &+2\lambda_2\gamma^2\|w_{\rm c}^T\phi_{\rm c}(t)\|^2\\ &+\lambda_2\gamma^2\left\|\xi_{\rm a}(t)C(t)-\hat{w}_{\rm c}^T(t)\phi_{\rm c}(t)\right\|^2\\ &+\lambda_2\gamma^2(1+2\alpha)\left\|\sum_{j=1}^{l}\phi_{a_j}(t)C_j(t)e_{a_j}^T(t)\right\|^2 \end{aligned}\tag{35}$$

With Assumption 1 and Cauchy–Schwarz inequality, we derive from (35)

$$\begin{aligned}\mathfrak{L}^2(t) \le& 3\lambda_1\left[2l^2+\beta\gamma^2(1+6l^2+\gamma^2)\right]w_{\rm cm}^2\phi_{\rm cm}^4\\ &+\left[3\lambda_1+\gamma^2(3\lambda_1+4\lambda_2)\right.\\ &\left.+\lambda_2\gamma^2l^2(1+2\alpha)\phi_{\rm am}^2C_m^2\right]w_{\rm cm}^2\phi_{\rm cm}^2\\ &+3\lambda_1\left[1+l^2+\beta\gamma^2(\phi_{\rm cm}^2+9l^2)\right]r_m^2+\lambda_1\gamma^2w_{\rm cm}^2\\ &+\gamma^2(\lambda_1+2\lambda_2C_m^2\phi_{\rm am}^2)w_{\rm am}^2\triangleq\mathfrak{L}_m^2 \end{aligned}\tag{36}$$

where r_m and C_m are the upper bound of $\|r(t)\|$ and $\|C(t)\|$, respectively, that is $\|r\|\le r_m$, $\|C(t)\|\le C_m$.

Select the parameters to satisfy that

$$0<\alpha\|C(t)\|^2\|\phi_{\rm a}(t)\|^2<1,\quad 0<\beta\gamma^2\|\phi_{\rm c}(t)\|^2<1\tag{37}$$

and the parameters $\lambda_i>0$ $(i=1,2)$ are chosen as

$$\frac{\lambda_1}{\lambda_2}>\frac{2}{1-\beta\gamma^2\|\phi_{\rm c}(t)\|^2}\tag{38}$$

Then, if (37) and (38) hold, for any

$$\|\xi_{\rm c}(t)\|>\sqrt{\frac{\mathfrak{L}_m^2}{\gamma^2\left[\lambda_1\left(1-\beta\gamma^2\|\phi_{\rm c}(t)\|^2\right)-2\lambda_2\right]}}\tag{39}$$

$$\|\xi_{\rm a}(t)\|>\sqrt{\frac{\mathfrak{L}_m^2}{\lambda_2\gamma^2\|C(t)\|^2}}\tag{40}$$

the first difference $\Delta L(t)<0$ holds.

Note that $\|\xi_{\rm c}(t)\|\le\|\tilde{w}_{\rm c}(t)\|\|\phi_{\rm cm}\|$ and $\|\xi_{\rm a}(t)\|\le\|\tilde{w}_{\rm a}(t)\|\|\phi_{\rm am}\|$, then by (39) and (40), we have

$$\|\tilde{w}_{\rm c}(t)\|>\frac{1}{\phi_{\rm cm}}\sqrt{\frac{\mathfrak{L}_m^2}{\gamma^2\left[\lambda_1\left(1-\beta\gamma^2\|\phi_{\rm c}(t)\|^2\right)-2\lambda_2\right]}}\triangleq\mathfrak{B}_{\rm c}$$

$$\|\tilde{w}_{\rm a}(t)\|>\frac{1}{\phi_{\rm am}}\sqrt{\frac{\mathfrak{L}_m^2}{\lambda_2\gamma^2\|C(t)\|^2}}\triangleq\mathfrak{B}_{\rm a}$$

With the standard Lyapunov extension theorem, we can conclude that the weight estimation errors of the critic network $\tilde{w}_{\rm c}(t)$ and the actor network $\tilde{w}_{\rm a}(t)$ are UUB by positive constants $\mathfrak{B}_{\rm c}$ and $\mathfrak{B}_{\rm a}$.

The proof is completed.

References

Adam S, Busoniu L, Babuska R (2012) Experience replay for real-time reinforcement learning control. IEEE Trans Syst Man Cybern C 42(2):201–212

Busoniu L, Babuska R, De Schutter B, Ernst D (2010) Reinforcement learning and dynamic programming using function approximators, vol 39. CRC Press, Boca Raton

Chowdhary G, Johnson E (2010) Concurrent learning for convergence in adaptive control without persistency of excitation. In: 2010 49th IEEE conference decision and control (CDC), pp 3674–3679

Dierks T, Thumati B, Jagannathan S (2009) Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. Neural Netw 22(5):851–860

dSPACE (2015) dSPACE ASM vehicle dynamics. http://www.dspace.com/en/inc/home/products/sw/automotive_simulation_models/produkte_asm/vehicle_dynamics_models.cfm. Accessed Oct 28, 2015

Grondman I, Busoniu L, Lopes GA, Babuska R (2012) A survey of actor-critic reinforcement learning: standard and natural policy gradients. IEEE Trans Syst Man Cybern C 42(6):1291–1307

Guvenc B, Kural E (2006) Adaptive cruise control simulator: a low-cost, multiple-driver-in-the-loop simulator. IEEE Control Syst Mag 26(3):42–55

Kang F, Han S, Salgado R, Li J (2015) System probabilistic stability analysis of soil slopes using gaussian process regression with latin hypercube sampling. Comput Geotech 63:13–25

Kang F, Xu Q, Li J (2016) Slope reliability analysis using surrogate models via new support vector machines with swarm intelligence. Appl Math Model 40(11):6105–6120

Kiumarsi B, Lewis FL (2015) Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. IEEE Trans Neural Netw Learn Syst 26(1):140–151

Kyongsu Y, Ilki M (2004) A driver-adaptive stop-and-go cruise control strategy. In: 2004 IEEE International Conference on Network Sensor Control, pp 601–606

Lin LJ (1992) Self-improving reactive agents based on reinforcement learning, planning and teaching. Mach Learn 8(3–4):293–321

Liu D, Wang D, Zhao D, Wei Q, Jin N (2012a) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. IEEE Trans Autom Sci Eng 9(3):628–634

Liu F, Sun J, Si J, Guo W, Mei S (2012b) A boundedness result for the direct heuristic dynamic programming. Neural Netw 32:229–235

Mnih V, Kavukcuoglu K, Silver D, Rusu et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533

Modares H, Lewis FL, Naghibi-Sistani MB (2014) Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. Automatica 50(1):193–202

Moon S, Moon I, Yi K (2009) Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance. Control Eng Pract 17(4):442–455

Rummery GA, Niranjan M (1994) On-line Q-learning using connectionist systems. Technical report CUED/F-INFENG/TR 166, Department of Engineering, University of Cambridge

Shakouri P, Ordys A (2014) Nonlinear model predictive control approach in design of adaptive cruise control with automated switching to cruise control. Control Eng Pract 26:160–177

Si J, Wang YT (2001) Online learning control by association and reinforcement. IEEE Trans Neural Netw 12(2):264–276

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT press, Cambridge

Tapani A (2012) Vehicle trajectory effects of adaptive cruise control. J Intell Transp Syst 16(1):36–44

Tsai CC, Hsieh SM, Chen CT (2010) Fuzzy longitudinal controller design and experimentation for adaptive cruise control and stop&go. J Intell Robot Syst 59(2):167–189

Wang B, Zhao D, Li C, Dai Y (2015) Design and implementation of an adaptive cruise control system based on supervised actor-critic learning. In: 2015 5th international conference on information science technology (ICIST), pp 243–248

Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292

Wawrzyński P, Tanwani AK (2013) Autonomous reinforcement learning with experience replay. Neural Netw 41:156–167

Xiao L, Gao F (2011) Practical string stability of platoon of adaptive cruise control vehicles. IEEE Trans Intell Transp Syst 12:1184–1194

Yang X, Liu D, Wang D, Wei Q (2014) Discrete-time online learning control for a class of unknown nonaffine nonlinear systems using reinforcement learning. Neural Netw 55:30–41

Zhang Q, Zhao D, Zhu Y (2016) Event-triggered H∞ control for continuous-time nonlinear system via concurrent learning. IEEE Trans Syst Man Cybern Syst 47(7):1071–1081

Zhao D, Bai X, Wang F, Xu J, Yu W (2011) DHP for coordinated freeway ramp metering. IEEE Trans Intell Transp Syst 12(4):990–999

Zhao D, Zhang Z, Dai Y (2012) Self-teaching adaptive dynamic programming for go-moku. Neurocomputing 78(1):23–29

Zhao D, Wang B, Liu D (2013) A supervised actor-critic approach for adaptive cruise control. Soft Comput 17(11):2089–2099

Zhao D, Xia Z, Wang D (2015a) Model-free optimal control for affine nonlinear systems with convergence analysis. IEEE Trans Autom Sci Eng 12(4):1461–1468

Zhao D, Zhang Q, Wang D, Zhu Y (2015b) Experience replay for optimal control of nonzero-sum game systems with unknown dynamics. IEEE Trans Cybern 46(3):854–865