Eligibility traces and forgetting factor in recursive least-squares-based temporal difference

Baldi, Simone; Zhang, Zichen; Liu, Di

DOI
10.1002/acs.3282

Publication date
2022

Document Version
Final published version

Published in
International Journal of Adaptive Control and Signal Processing

Received: 29 December 2020 | Revised: 10 February 2021 | Accepted: 9 May 2021

DOI: 10.1002/acs.3282

SPECIAL ISSUE ARTICLE

Eligibility traces and forgetting factor in recursive least-squares-based temporal difference

Simone Baldi[1,2] | **Zichen Zhang**[1] | **Di Liu**[3,4]

[1]School of Mathematics, Southeast University (SEU), Nanjing, China

[2]Delft Center for Systems and Control (DCSC), Delft University of Technology (TU Delft), Delft, The Netherlands

[3]School of Cyber Science and Engineering, Southeast University (SEU), Nanjing, China

[4]Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen (RUG), Groningen, The Netherlands

Correspondence
Di Liu, School of Cyber Science and Engineering, Southeast University (SEU), Jiulonghu Campus, Nanjijng 211189, China.
Email: liud923@126.com

Funding information
Jiangsu Double Innovation Talents Project for Jiangsu province, Grant/Award Number: 4207012004; National Natural Science Foundation of China, Grant/Award Number: 62073074

Summary

We propose a new reinforcement learning method in the framework of Recursive Least Squares-Temporal Difference (RLS-TD). Instead of using the standard mechanism of eligibility traces (resulting in RLS-TD(λ)), we propose to use the forgetting factor commonly used in gradient-based or least-square estimation, and we show that it has a similar role as eligibility traces. An instrumental variable perspective is adopted to formulate the new algorithm, referred to as RLS-TD with forgetting factor (RLS-TD-f). An interesting aspect of the proposed algorithm is that it has an interpretation of a minimizer of an appropriate cost function. We test the effectiveness of the algorithm in a Policy Iteration setting, meaning that we aim to improve the performance of an initially stabilizing control policy (over large portion of the state space). We take a cart-pole benchmark and an adaptive cruise control benchmark as experimental platforms.

KEYWORDS

eligibility traces, instrumental variable method, least squares, reinforcement learning, temporal difference

1 | INTRODUCTION

Reinforcement learning has received increasing attention not only in the machine learning field,[1] but also in the systems and control field,[2,3] and in the complex optimization and decision-making fields.[4-6] Several reinforcement learning methods have been proposed in literature, with temporal difference (TD) learning being one of the most popular. TD dates back to the 80's, and was proposed by Sutton as a way to estimate the vale function of a Markov Decision Process (MDP).[7] The original method was called TD(0): an extension of it was called TD(λ), where the parameter $0 < \lambda < 1$ is associated to the mechanism of *eligibility traces*. The TD(λ) algorithm was originally proposed for Markov Decision Process, which are problems with finite state and action spaces. However, motivated by the fact that many technological applications exhibit large state space, *value function approximation* methods have also been designed. The term 'value function approximation' refer to the fact that instead of describing the value function as a table evaluated for different states, a parametrized description of the value function is used via approximators such as linear regression or neural networks. Depending on its structure, the function approximator can be linear-in-the-parameters or nonlinear-in-the-parameters: when combined with nonlinear-in-the-parameters value function approximators (also called nonlinear value function approximation for brevity, or also nonlinear TD(λ)), TD(λ) cannot guarantee convergence in general.[8] However, for TD(λ) with linear-in-the-parameters function approximators (also called linear value function approximation for brevity, or also linear TD(λ)), convergence was shown by Dayan.[9]

Few years after the TD algorithm, by using the theory of linear least-squares estimation, Bradtke and Barto proposed two temporal difference algorithms, called Least-Squares TD(0) (LS-TD(0)) and Recursive Least-Squares TD(0) (RLS-TD(0)), respectively.[10] The main difference between LS-TD(0) and RLS-TD(0) is whether the parameter of the value function approximator is computed from a batch of data, or updated every time a new data arrives.[11] Later, Boyan proposed a class of linear temporal difference learning algorithms called LS-TD(λ),[12] where the mechanism of eligibility traces was embedded in the least-squares estimation. Analogously, the mechanism of eligibility traces can be included in RLS-TD(0), resulting in the RLS-TD(λ) algorithm, proposed and analyzed in Reference 13.

It is worth remarking that the TD error is the basic block of many reinforcement learning methods, including the well-known Q-learning and SARSA.[14,15] Many other examples of algorithms relying on the TD method have appeared in literature. A not-exhaustive list is provided as follows: natural gradient descent applied to Bellman error formulation was investigated in Reference 16; an algorithm named Complementary Temporal Difference Learning (CTDL) was described in Reference 17, which is based on the combination of deep neural networks with self-organized mapping updated by the TD error; a learning algorithm named graph Laplacian dual heuristic programming (GL-DHP) was proposed in Reference 18, which is based on the integration of manifold learning methods with adaptive critic networks; other methods can also be found in References 19-21 and references therein. It is also worth mentioning that in the recent decade the connections between reinforcement learning and adaptive optimal control have been more extensively explored: the connection amounts to formulating value iteration or policy iteration problems as estimation problems of some optimal parametrized value function or some optimal parametrized policy function.[22-28] While several methods exist, they all rely on some sort of TD error in order to update the parameter estimators.

Therefore, it is crucial to provide possible extensions or new insights into the TD method. A well-recognized open problem in TD concerns the mechanism of eligibility traces, since a clear interpretation to mechanism is still unclear, and even the selection of the parameter of the eligibility trace is unclear.[1] In this work, we show that the mechanism of eligibility traces is analogous to the use of forgetting factor in parameter estimation methods (such as gradient-based or least-squares estimation).

Considering the gradient structure of TD algorithm, we propose a new TD algorithm that we name RLS-TD with forgetting factor (abbreviated as RLS-TD-f). This new algorithm comes from a new insight into TD, which is formulated as an instrumental-variable method combined with an online optimization. An interesting aspect of the proposed algorithm is that it has an interpretation of a minimizer of an appropriate cost function. A preliminary version of this work has been presented for conference.[29]

The paper is organized as follows. In Section 2, we recall important concepts of reinforcement learning. Section 3 illustrates the following algorithms: TD(λ), LS-TD(λ), and RLS-TD(λ), all of them using linear function approximators. In Section 4, we introduce an instrumental variable method to the purpose of providing a new optimization-based perspective of RLS-TD. In Section 5, we test the effectiveness of the proposed idea on some control benchmarks.

2 | OVERVIEW OF REINFORCEMENT LEARNING

Reinforcement learning is traditionally formulated for Markov Decision Processes (MDPs), a framework to describe stochastic optimal control with finite state space and finite action space.[2] Let us recall some concepts from the reinforcement learning problem, that is, the closed-loop control problem of mapping states (feedback signals) into actions to maximize a reward signal (cf. Figure 1).

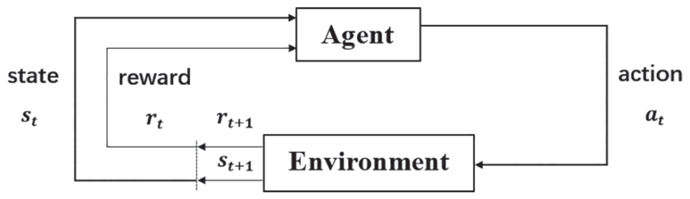


FIGURE 1 Interaction between agent and environment in reinforcement learning

At each time step $t = 0, 1, 2, 3, \ldots$, let the agent (an 'agent' is nothing but a controller) receive some environmental state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, and a reward signal $r_{t+1} \in \mathcal{R}$. With this information, the agent selects an action $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ is the set of actions available in state s_t and the situation evolves into a new state s_{t+1} and a new reward $r_{t+1} \in \mathcal{R}$ (which are in part the consequence of the action a_t).

Four main ingredients are present in a reinforcement learning system: a policy, a reward, a value function (or an action value function), and a model of the environment.

A *policy* defines the way an agent behaves, through a mapping from the state space to the action space.

A *reward signal* defines the goal, through a mapping from the agent's current action space and the current state space of the agent's environment to the reward space. The agent's goal is to maximize a cumulative reward

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1}$$

where $0 < \gamma \leq 1$, called the discount rate to give less importance to rewards far in the future. Note that the agent's actions have a direct effect on reward, and an indirect effect through changing the environment's state.

The *value function* $V_\pi(s)$ is a crucial function in reinforcement learning since it specifies, for a given state s and following a certain policy π, the total amount of rewards an agent can expect to accumulate over the future

$$V_\pi(s) = E_\pi[G_t | S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s] \tag{2}$$

where $E_\pi[\cdot]$ denotes the expected value given the policy π. The value function is defined with respect to a particular policy, because the rewards the agent can expect to receive depend on what actions it will take. Similarly, we define the *action value function* $Q_\pi(s, a)$ as the value of taking action a in state s under a policy π

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[G_t | S_t = s, A_t = a] \\ &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a]. \end{aligned} \tag{3}$$

Finally, the *model of the environment* describes the evolution of the state as the consequence of taking a certain action, through a mapping from the current space and current action to the next state.

A fundamental property of value functions is to satisfy the Bellman dynamic programming recursions

$$\begin{aligned} V_\pi(s) &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a)[r + \gamma V_\pi(s')] \end{aligned} \tag{4}$$

where

$$p(s', r | s, a) = Pr\{S_{t+1} = s', r_{t+1} = r | S_t = s, A_t = a\}.$$

Solving a reinforcement learning problem amounts to finding the policy π^* giving the best cumulative reward, that is,

$$V_*(s) = \max_\pi V_\pi(s), \quad \forall s \in \mathcal{S}. \tag{5}$$

or equivalently

$$Q_*(s, a) = \max_\pi Q_\pi(s, a) \tag{6}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(\int)$. For the state–action pair (s, a), the function $Q_*(s, a)$ gives the expected return for taking action a in state s and thereafter following the optimal policy. Thus, we can write Q_* in terms of V_*

$$Q_*(s, a) = E[r_t + \gamma V_*(S_{t+1}) | S_t = s, A_t = a]. \tag{7}$$

Researchers are interested in optimal value functions (or optimal action value functions) rather than optimal policies because an optimal policy can be calculated from V_* via

$$\begin{aligned} V_*(s) &= \max_a E[r_{t+1} + \gamma V_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r | s, a)[r + \gamma V_*(s')]. \end{aligned} \tag{8}$$

or equivalently by finding any action that maximizes $Q_*(s, a)$.

3 | OVERVIEW OF TEMPORAL DIFFERENCE

The temporal difference (TD) was originally proposed in Reference 7 as a method to estimate a value function. TD calculates the value of the state through the value of the next state in an iterative way, the so-called TD(0) formula

$$\begin{aligned} V(s_t) &\leftarrow V(s_t) + \alpha \delta_t \\ \delta_t &= r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{9}$$

where δ_t is called TD error, α is a positive learning step, γ is the aforementioned discount rate of the cumulative reward.

For MDPs, the value function $V(s_t)$ is essentially stored as a table of the state. Unfortunately, MDPs stop being very effective when the dimension of the state space becomes larger and larger, since in this case the table becomes large and larger. When the state space is large (which is often the case in practical optimal control problems), function approximators can be used to approximate the value function (a typical type of function approximators are linear function approximators).[13,30] This implies that, instead of storing the value function as a table of the state, the value function has a parametrized form

$$V_t(s) = \phi_t(s)^T \theta_t \tag{10}$$

where $\theta_t = (\theta_1, \theta_2, \ldots, \theta_K)^T$ is the weight vector and $\phi(s) = (\phi_1(s), \phi_2(s), \ldots \phi_K(s))^T$ is the basis function space of the linear function approximator (with dimension K). When adopting linear function approximators, the TD(0) learning rule (9) becomes

$$\theta_{t+1} = \theta_t + \alpha(r_{t+1} + \gamma \phi_{t+1}^T \theta_t - \phi_t^T \theta_t)\phi_t. \tag{11}$$

A standard interpretation of TD methods is via the gradient descent method (or steepest descent method) in the space of the weights θ.[7] In other words, TD can be seen as a minimizer for the error measure $J(\theta)$

$$J(\theta_t) = \sum_{k=1}^{t} [r_k - (\phi_k^T - \gamma \phi_{k+1}^T)\theta]^2. \tag{12}$$

The weight vector θ_t is iteratively updated along the direction in which $J(\theta)$ decreases most steeply, so the error measure can be eventually minimized: (11) can be written as

$$\theta_{t+1} = \theta_t + \alpha \Delta\theta_t, \quad \Delta\theta_t = -\alpha \bigtriangledown_\theta J(\theta_t) \tag{13}$$

where $\bigtriangledown_\theta J(\theta)$ is the gradient of $J(\theta)$ with respect to theta, which is indeed a steepest descent method.

Algorithm 1. TD(λ):

```
Given: (1) MDP or simulation model with policy $\pi$ defined in $\mathcal{S}$;
       (2) a featurizer $\Phi : \mathcal{S} \rightarrow \Re^K$ mapping states
       to feature vectors, $\Phi(END) \overset{def}{=} 0$;
       (3) a parameter $\lambda \in [0,1]$;
       (4) learning step size: $\alpha$;
Output: vector $\theta$ so that $V_\pi(s) \approx \phi^T(s)\theta$.
Set initial estimate $\theta$ for $t := 0$.for n:=1,2,... do
    Set $\delta$:=0.
    Starting state $s_t \in \mathcal{S}$.
    Set $e_t := \phi(s_t)$.
    while $s_t \neq END$, do
        Simulate MDP for one step, obtain reward $r_t$ and next state $s_{t+1}$.
        Set $\delta := \delta + (r_t + (\gamma\phi(s_{t+1}) - \phi(s_t))^T\theta)$.
        Set $e_{t+1} := \gamma\lambda e_t + \phi(s_{t+1})$.
        Update $t := t + 1$.
    Update $\theta := \theta + \alpha\delta e$
```

3.1 | The eligibility trace mechanism

Eligibility traces are a basic mechanism of reinforcement learning.[31] Although there is no formal interpretation of this mechanism, a popular interpretation[1] is that the eligibility trace records of the occurrence of an event. According to this interpretation, the trace makes the parameters associated with the event as eligible for being updated, according to the TD error.

The standard temporal difference algorithm TD(0), when combined with eligibility trace, results in the TD(λ) algorithm:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha(r_{t+1} + \gamma\phi_{t+1}^T\theta_t - \phi_t^T\theta_t)e_t,\\ e_t &= \gamma\lambda e_{t-1} + \phi_t\end{aligned} \tag{14}$$

where $\lambda \in [0,1]$ is the eligibility trace parameter. When $\lambda = 1$, the change of each state in the past has a complete impact on updating θ_{t+1}, as in Monte Carlo method; when $\lambda = 0$, the change of each state in the past has no impact on updating θ_{t+1}, as in TD(0); when $\lambda \in (0,1)$, each state in the past has an impact on updating θ_{t+1} in a declining trend.

Algorithm 1 shows TD(λ) with linear function approximator.[8]

The TD(λ) algorithm can be roughly illustrated as follows: on each iteration, the TD(λ) algorithm computes the one-step TD error $r_t + (\gamma\phi(s_{t+1}) - \phi(s_t))^T\theta_t$, and uses this error for updating all state features according to their respective eligibility trace e_t.

3.2 | Least squares combined with eligibility traces

Motivated by some shortcomings of TD(λ), Boyan[12] proposed an improved method, namely Least-Squares TD(λ) (LS-TD(λ)): its main feature is to make use of an experience matrix A (of dimension $K\times K$, where K is the dimension of the basis function space) and builds a vector b (of dimension K). These are built as

$$b = \sum_{i=0}^{L} e_i r_i \qquad A = \sum_{i=0}^{L} e_i(\gamma\phi(s_{i+1}) - \phi(s_i))^T. \tag{15}$$

After K collecting independent regressors (so that A is invertible), then, θ is estimated as $A^{-1}b$ (cf. Algorithm 2).

Algorithm 2. LS-TD(λ):

```
Given: (1) MDP or simulation model with policy $\pi$ defined in $\mathcal{S}$;
       (2) a featurizer $\Phi : \mathcal{S} \to \mathfrak{R}^K$ mapping states
       to feature vectors, $\Phi(END) \overset{def}{=} 0$;
       (3) a parameter $\lambda \in [0, 1]$;
Output: vector $\theta$ so that $V_\pi(s) \approx \phi^T(s)\theta$.
Set $A := 0$, $b := 0$, $t := 0$.
for $n:=1,2,\ldots$ do
    Starting state $s_t \in \mathcal{S}$.
    Set $e_t := \phi(s_t)$.

    while $s_t \neq END$, do
        Simulate MDP for one step, obtain reward $r_t$ and next state $s_{t+1}$.
        Set $A := A + e_t(\phi(s_t) - \gamma\phi(s_{t+1}))^T$.
        $b := b + e_t r_t$.
        Set $e_{t+1} := \gamma\lambda e_t + \phi(s_{t+1})$.
        Update $t := t + 1$.

    Desired coefficients are: Set $\theta := A^{-1}b$
```

When $\lambda = 0$, LS-TD(λ) reduces to LS-TD(0), derived by Bradtke and Barto using the theory of regression with instrumental variables.[10] As pointed out by Bradtke and Barto in Reference 10, least-squares methods offer several advantages:

- Least squares extract more information from each sample and typically converge with fewer samples.
- The convergence of TD(λ) can be slow if the step size α is chosen poorly. LS-TD(λ) eliminates the need to choose an appropriate α.
- The performance of TD(λ) is sensitive to the initial θ.

3.3 | Recursive least squares combined with eligibility traces

LS-TD(λ) requires inverting a matrix at each time step, which has computational complexity $O(K^3)$, assuming that the space of basis functions has dimension K. Recursive Least-Squares (RLS) have been used to derive RLS-TD(λ),[10] that has computational complexity of $O(K^2)$.

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})^T P_{t-1} e_t}\delta_t e_t \tag{16}$$

$$P_t = P_{t-1} - \frac{P_{t-1} e_t (\phi_t - \gamma\phi_{t+1})^T P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})^T P_{t-1} e_t} \tag{17}$$

$$\delta_t = r_t - (\phi_t - \gamma\phi_{t+1})^T \theta_{t-1} \tag{18}$$

where

$$e_{t+1} = \gamma\lambda e_t + \phi_{t+1}. \tag{19}$$

It is recognized that there are no formal rules (except from trial-and-error) for selecting the eligibility trace parameter λ.[32] Two views (a forward view and a backward view) have been proposed for the eligibility trace mechanism: in the forward view, the mechanism can be understood as a way of averaging n-step backups (i.e., a way in between a TD(0) method and a Monte Carlo method); in the backward view, the eligibility traces decay by $\lambda\gamma$ for all states, and is incremented by 1 for the one visited state (as a result, the TD error generates proportional updates to all recently visited states). The following section will show that the backward view of the eligibility trace mechanism can be formulated in terms of forgetting factor inside an appropriately designed online optimization problem.

4 | AN INSTRUMENTAL VARIABLE PERSPECTIVE OF RECURSIVE TD

4.1 | Gradient structure of TD update

Going back to the standard TD method (11) for a while, we have seen that the TD(0) can be written as a gradient update (13): interestingly,[7] showed that the learning rule for TD(λ) (i.e., with eligibility trace mechanism) can also be written as a gradient update

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha[r_t + \gamma V_t(s_{t+1}) - V_t(s_t)]\sum_{k=1}^{t}\lambda^{t-k}\nabla_{\theta_t}V_t(s_k)\\ &= \theta_t + \alpha \triangle \theta_t \end{aligned} \tag{20}$$

where

$$\triangle\theta_t = [r_t + \gamma V_t(s_{t+1}) - V_t(s_t)]\sum_{k=1}^{t}\lambda^{t-k}\nabla_{\theta_t}V_t(s_k). \tag{21}$$

Note that the summation in (21) keeps $\triangle\theta_t$ fixed, while spanning the recently visited states s_k. The literature suggest to do this to decouple the effect of changing the parameters from the effect of moving in state space.[10] Here comes or observation: interestingly, (20) exhibits a similar structure as the LS cost (12): in fact, in this cost the parameter θ_t is fixed while spanning the past data $\hat{\omega}_k$.[33] A natural question is therefore to find the corresponding version of (12) for the update (20)–(21).

We will investigate this question in the framework of instrumental variable regression, whose goal is to obtain a least-squares approximation of some function $R : \Re^n \to \Re$, given samples of $\omega_t \in \Re^n$ and outputs $r_t \in \Re^n$. In the standard least-square regression, only the observations r_t are corrupted by noise

$$r_t = \omega_t^T\theta^* + \eta_t \tag{22}$$

where θ^* is the vector of true but unknown parameters and η_t is the noise. It is well known that, in this case, the standard least-square regression comes from the minimization of (12) (define $\omega_t = \phi_t - \gamma\phi_{t+1}$), and has the following closed-form solution

$$\theta_t = \left[\sum_{k=1}^{t}\omega_k\omega_k^T\right]^{-1}\left[\sum_{k=1}^{t}\omega_k r_k\right] \tag{23}$$

obtained by setting the gradient of J_t with respect to θ_t equal to zero.

However, in instrumental variable regression, the input observations ω_t are also noisy, resulting in the following errors-in-variables situation[11]

$$r_t = \hat{\omega}_t\theta^* - \zeta_t^T\theta^* + \eta_t \tag{24}$$

where ζ_t is another observation noise. Substituting $\hat{\omega}_t$ directly for ω_t in (23) introduces a bias, which unfortunately prevents θ_t from converging to θ^*.[10,11] To solve this problem, it was proposed to introduce an instrumental variables ϕ_t, which is a vector correlated with ω_t, but uncorrelated with ζ_t. Then, (23) gets modified into

$$\theta_t = \left[\sum_{k=1}^{t}\phi_k\hat{\omega}_k^T\right]^{-1}\left[\sum_{k=1}^{t}\phi_k r_k\right]. \tag{25}$$

It was shown in Reference 10 that LS-TD(0) can be interpreted using the instrumental variable method with $\hat{\omega}_k = \phi_k - \gamma\phi_{k+1}$. As the next step, we will use the instrumental variable method to provide a new optimization perspective into LS-TD(0) and LS-TD(λ).

4.2 | The minimizer corresponding to instrumental variable method

In this section we want to derive RLS-TD(0) according to the instrumental variable method. This new perspective leads to an optimization cost for the RLS-TD(0) which has not appeared before in the TD literature.

Theorem 1. *The following online algorithm*

$$P_k = P_{k-1} - \frac{P_{k-1}\phi_k\hat{\omega}_k^T P_{k-1}}{m_k^2 + \hat{\omega}_k^T P_{k-1}\phi_k} \tag{26}$$

$$\theta_k = \theta_{k-1} + P_k\phi_k\left[\frac{r_k - \theta_{k-1}^T\hat{\omega}_k}{m_k^2}\right]. \tag{27}$$

gives the optimal solution to the cost function

$$J(\theta) = \frac{1}{2}(r_k - \Phi_k\theta)^T(r_k - \Phi_k\theta) - \frac{1}{2}(\Phi_k\theta)^T[(\Phi_k - \hat{\Omega}_k)\theta] + \frac{1}{2}(\theta - \theta_0)^T P_0^{-1}(\theta - \theta_0). \tag{28}$$

which has an interpretation in terms of the instrumental variables method (see proof).

Proof. We consider the following minimizer, which is motivated by LS-TD(0) in Reference 10

$$\theta = (\Phi_k^T\hat{\Omega}_k + P_0^{-1})^{-1}(P_0^{-1}\theta_0 + \Phi_k^T r_k) \tag{29}$$

where $\hat{\Omega}_k = [\hat{\omega}_1, \hat{\omega}_2, \dots, \hat{\omega}_k]$ is the regressor vector, $\Phi_k = [\phi_1, \phi_2, \dots, \phi_k]$ is the instrumental variable vector, $\hat{\omega}_k = \phi_k - \gamma\phi_k$, $r_k = [r_1, r_2, \dots, r_k]$ is the reward vector. The matrix P_0 and the vector θ_0 will be defined later.

We want to find the cost corresponding to the minimizer (29), that is, we want to find a cost function J such that the minimizer makes its gradient equal to 0

$$\begin{aligned}\bigtriangledown J(\theta) &= -\Phi_k^T(r_k - \hat{\Omega}_k\theta) + P_0^{-1}(\theta - \theta_0)\\ &= -\Phi_k^T r_k + \Phi_k^T\hat{\Omega}_k\theta + P_0^{-1}(\theta - \theta_0) = 0.\end{aligned} \tag{30}$$

From reverse calculation, we can get the following cost function

$$J(\theta) = \frac{1}{2}(r_k - \Phi_k\theta)^T(r_k - \Phi_k\theta) - \frac{1}{2}(\Phi_k\theta)^T[(\Phi_k - \hat{\Omega}_k)\theta] + \frac{1}{2}(\theta - \theta_0)^T P_0^{-1}(\theta - \theta_0). \tag{31}$$

The next step is as follows: we want to avoid using (30) to calculate the minimizer, and we replace it with a recursive method that updates the minimizer when new data arrive.

Define $P^{-1} = \Phi_k^T\hat{\Omega}_k + P_0^{-1}$, then

$$P_k^{-1} - P_{k-1}^{-1} = \Phi_k^T\hat{\Omega}_k - \Phi_{k-1}^T\hat{\Omega}_{k-1} = \frac{\phi_k^T\hat{\omega}_k}{m_k^2} \tag{32}$$

where $m_k^2 \geq 1$ is a normalizing signal to bound ϕ_k from above.

$$P_k^{-1} = P_{k-1}^{-1} + \frac{\phi_k^T\hat{\omega}_k}{m_k^2} \tag{33}$$

Using the matrix inversion lemma[11]

$$(A + BC)^{-1} = A^{-1} + A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1},$$

with

$$A = P_{k-1}^{-1}, B = \frac{\phi_k}{m_k}, C = \frac{\hat{\omega}_k^T}{m_k}$$

we get

$$P_k = P_{k-1} - \frac{P_{k-1}\phi_k\hat{\omega}_k^T P_{k-1}}{m_k^2 + \hat{\omega}_k^T P_{k-1}\phi_k} \tag{34}$$

$$\begin{aligned}\theta_k &= P_k(P_0^{-1}\theta_0 + \Phi_k^T r_k) \\ &= P_k(P_0^{-1}\theta_0 + \Phi_{k-1}^T r_{k-1} + \frac{\phi_k r_k}{m_k^2}),\end{aligned} \tag{35}$$

note that $(\Phi_{k-1}^T \hat{\Omega}_{k-1} + P_0^{-1})\theta = P_0^{-1}\theta_0 + \Phi_{k-1}^T r_k$, which gives us the recursive formulas

$$\theta_k = P_k\left(P_k^{-1}\theta_{k-1} - \frac{\phi_k\hat{\omega}_k\theta_{k-1}}{m_k^2} + \frac{\phi_k r_k}{m_k^2}\right) \tag{36}$$

$$\theta_k = \theta_{k-1} + P_k\phi_k\left[\frac{r_k - \theta_{k-1}^T\hat{\omega}_k}{m_k^2}\right]. \tag{37}$$

■

4.3 | Proposed method

Let us now follow a similar approach as in Theorem 1, but using a different cost.

Theorem 2. *The following online algorithm*

$$P_k = \frac{1}{\beta}\left[P_{k-1} - \frac{a_k P_{k-1}\phi_k\hat{\omega}_k^T P_{k-1}}{m_k^2\beta + a_k\hat{\omega}_k^T P_{k-1}\phi_k}\right] \tag{38}$$

$$\theta_k = \theta_{k-1} + \sqrt{a_k}P_k\phi_k\left[\frac{r_k - \theta_{k-1}^T\hat{\omega}_k}{m_k^2}\right]. \tag{39}$$

gives the optimal solution to the cost function

$$J(\theta) = \frac{1}{2}\sum_{k=1}^{t} a_k\beta^{t-k}\frac{[r_k - \theta_k^T\phi_k]^2}{m_k^2} - \frac{1}{2}\sum_{k=1}^{t} a_k\beta^{t-k}\frac{(\phi_k\theta_k^T)^T[(\phi_k - \hat{\omega}_k)\theta_k^T]}{m_k^2} + \frac{1}{2}\beta^t(\theta_k - \theta_0)^T P_0^{-1}(\theta_k - \theta_0) \tag{40}$$

where a_k are non-negative weighting coefficients, and $\beta \in (0, 1)$.

Proof. Vector θ_k, the estimate of θ^*, is obtained by solving

$$\nabla J(\theta) = \beta^t P_0^{-1}(\theta_k - \theta_0) - \sum_{k=1}^{t} a_k\beta^{t-k}\frac{r_k - \theta_k^T\phi_k}{m_k^2}\phi_k - \frac{1}{2}\sum_{k=1}^{t} a_k\beta^{t-k}\frac{\phi_k^T(\phi_k - \hat{\omega}_k\theta_k^T) + (\phi_k\theta_k^T)^T(\phi_k - \hat{\omega}_k)}{m_k^2} = 0. \tag{41}$$

Similar to the derivation process of RLS-TD(0), we can get that

$$\theta_k = P_k(\beta^t P_0^{-1}\theta_0 + a_k\beta^{t-k}\Phi_{k-1}^T r_{k-1} + a_k\beta^{t-k}\frac{\phi_k r_k}{m_k^2}). \tag{42}$$

$$P_k = \frac{1}{\beta}\left[P_{k-1} - \frac{a_k P_{k-1}\phi_k\hat{\omega}_k^T P_{k-1}}{m_k^2\beta + a_k\hat{\omega}_k^T P_{k-1}\phi_k}\right] \tag{43}$$

The equation for θ_k may also be written in the form

$$\theta_k = \theta_{k-1} + \sqrt{a_k} P_k \phi_k \left[\frac{r_k - \theta_{k-1}^T \hat{\omega}_k}{m_k^2} \right]. \tag{44}$$

■

The main differences between RLS-TD(λ) and RLS-TD-f can be listed as follows: the forgetting factor introduced in the cost (40) reduces the influence of historical data. Qualitatively, this is a similar mechanism in (20)–(21) which was used to introduce the eligibility trace mechanism. In other words, cost (40) is a generalized version of cost (12) used to introduce the eligibility trace mechanism. The similarities are in the use of the same one step error and in the fact of updating both the weight vector and the covariance matrix. The differences are in the update of the covariance matrix. Another difference is that in the proposed RLS-TD-f(λ) we make use of P_t to update θ_t, which leads to a different update as compared to RLS-TD(λ).

4.4 | The TD error in SARSA and Q-Learning

The control problem via TD method can be divided into two categories. One is online control (or on-policy), that is, a control policy is used to both update the value function and select new actions. The other is offline control (or off-policy), which uses two control policies, one for selecting new actions and the other for updating the value function.

More formally, we can define

- Target policy: the policy used for learning and training;
- Behavior policy: the policy used to interact with the environment to generate data, that is, to make decisions in the training process.

If at every time step the target policy coincides with the behavior policy, the learning is called on-policy learning. Otherwise it is off-policy learning, that is, improve a policy different from that used to generate the data. The most celebrated on-policy method is SARSA, and the most celebrated off-policy method is Q-Learning. Both are recalled below.

On-policy—SARSA: The objective is to learn an action-value function $Q(s, a)$ rather than $V(s)$. The price to pay is representing a function of state–action pairs (instead of just of states), but the advantage is to select optimal actions without the need to know the next state, that is, without a model of the environment's dynamics. The generalization of the TD(0) formula (9) in this scenario is straightforward:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{45}$$

The name SARSA comes from the sequence of State, Action, Reward, State, Action inside the algorithm. One common problem of on-policy and off-policy algorithms is that they may not explore extensively the search space, possibly giving a suboptimal answer or possibly getting stuck in its current policy. To address this issue, it is common to use the so-called ϵ-greedy policy approach. This amounts to selecting a different action with a small probability ϵ. This is formalized as follows

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon, & \text{if } a^* = \arg\max_{a \in A} Q(s, a) \\ \epsilon/m, & \text{else} \end{cases} \tag{46}$$

which means selecting the action corresponding to the best action-value function with probability 1 - ϵ, and using a different action with probability ϵ.

Algorithm 3. SARSA(λ):

```
Initialize: initial $Q(s,a)$, $e(s,a) = 0\ \forall s, a$
repeat
    Starting $s, a$
    repeat
        Apply action $a$, get $r, s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        calculate $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
        update $e(s,a) \leftarrow e(s,a) + 1$
        for all $s, a$: do
            update $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
            update $e(s,a) \leftarrow \gamma\lambda e(s,a)$
        update $s \leftarrow s'; a \leftarrow a'$
    until :
        $s$ is terminal
until :
    end
```

Similar to TD(λ), on-policy and off-policy algorithms can both make use of the eligibility trace mechanism. For example, SARSA(λ) can be obtained as shown below

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\delta_t e_t \tag{47}$$

where $\delta_t = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$, and e_t is the eligibility trace. See Algorithm 3.

Off-policy—Q-Learning: It is defined by the update rule:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \tag{48}$$

The main difference here is that instead of using $\gamma Q(s_{t+1}, a_{t+1})$ as in SARSA algorithm, where a_{t+1} is picked from an ε-greedy policy, in Q-Learning we pick the actual optimal action according to $max_a Q(s_{t+1}, a)$. Also for Q-Learning, one can use an ε-greedy method to select new actions. But for the update of value function, Q-Learning uses the greedy method $\max_a Q(s_{t+1}, a)$, without ε-greedy approach. Algorithm 4 shows the Q-Learning(λ) which makes use of the eligibility trace mechanism.

It is clear that the LS and RLS versions of SARSA and Q-Learning can be obtained in a similar way as LS-TD(λ) and RLS-TD(λ). As in TD algorithm, this requires to introduce linear function approximators which can be expressed (for both SARSA and Q-Learning) as

$$Q_t(s,a) = \phi_t^T(s,a)\theta_t, \tag{49}$$

where $\phi_t^T(s,a)$ (instead of $\phi_t^T(s)$) refers to the feature-extraction function that maps state s and action a to valued feature (column) vector with K dimensions, and θ_t has a similar meaning as the previously introduced vector, except having a different dimension. Accordingly, one can follow a similar instrumental variable procedure and obtain the versions of SARSA and Q-Learning using the proposed forgetting factor method. This is left to the reader to avoid repetitions.

An important aspect of reinforcement learning has not been addressed in this work, which is the so-called exploitation-exploration dilemma. This dilemma amounts to the problem of introducing or not probing noise in the control action in an effort to better estimate the optimal value or action value function. Interestingly, this problem is connected to the issue of persistence of excitation in adaptive control, as it was shown in the framework of adaptive optimization.[34-36] Because recently some approaches have been proposed in adaptive control about estimation with reduced persistence of excitation,[37,38] it is reasonable to expect that these ideas can be beneficial also in the reinforcement learning field.

Algorithm 4. Q-Learning(λ):

```
Initialize: initial $Q(s,a)$, $e(s,a) = 0\ \forall s, a$
repeat
    Starting $s, a$
    repeat
        Apply $a$, get $r, s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $a^* \leftarrow \arg\max_b Q(s', b)$
        calculate $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
        update $e(s,a) \leftarrow e(s,a) + 1$
        for all $s, a$: do
            update $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
            If $a' = a^*$, then $e(s,a) \leftarrow \gamma\lambda e(s,a)$
            else $e(s,a) \leftarrow 0$
        update $s \leftarrow s'$; $a \leftarrow a'$
    until :
        $s$ is terminal
until :
    end
```

5 | VALIDATION

5.1 | Cart-pole balancing

The balancing problem of inverted pendulums is a typical nonlinear control problem, studied by several communities in control theory and in artificial intelligence.[1] Figure 2 shows the cart-pole balancing scenario, which consists of a cart moving horizontally on a rail and a pole (the inverted pendulum) that has one end attached to the cart. The system can be presented by two variables (and their corresponding velocities): x represents the horizontal position of the cart, while ϑ represent the angle of the pole with respect to the vertical axis (note that $\vartheta = 0$ is the downright position and $\vartheta = \pi$ is the desired upright position). The control input is F, representing the horizontal force applied to the cart. The behavior of the cart-pole system can be described by the differential equations

$$\ddot{x} = \frac{1}{m_c + m_p \sin^2\vartheta}[F + m_p \sin\vartheta(l\dot{\vartheta}^2 + g\cos\vartheta)] \tag{50}$$

$$\ddot{\vartheta} = \frac{1}{l(m_c + m_p \sin^2\vartheta)} \times [-F\cos\vartheta - m_p l\dot{\vartheta}^2\cos\vartheta\sin\vartheta - (m_c + m_p)g\sin\vartheta] \tag{51}$$

where g is the acceleration due to the gravity, that is, $9.81 m/s^2$.

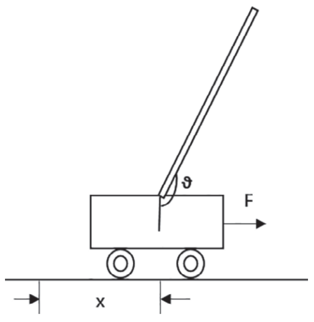


FIGURE 2 The cart-pole balancing control system

According to the values suggested in Reference 1, we set the mass of the cart $m_c = 1.0$ kg, the mass of the pole $m_p = 0.1$ kg, the pole length $l = 0.5$ m. The reward for the problem is defined as

$$r = -x^2 - \vartheta^2 - \dot{x}^2 - \dot{\vartheta}^2 - \rho F^2 \tag{52}$$

which is a quadratic cost. Here, ρ is a design constant representing the importance of the input in the cost (in order to avoid applying large inputs). In the simulation experiment, we set $\rho = 0.1$. Furthermore, we select the following approximator, which is motivated by a quadratic value function which is zero at the desired equilibrium

$$\phi_t(s) = [x_t^2, \dot{x}_t^2, (\vartheta_t - \pi)^2, \dot{\vartheta}_t^2, x_t\dot{x}_t, x_t(\vartheta_t - \pi), x_t\dot{\vartheta}_t, \dot{x}_t(\vartheta_t - \pi), \dot{x}_t\dot{\vartheta}_t, (\vartheta_t - \pi)\dot{\vartheta}_t]^T \tag{53}$$

$$\theta_t = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}]^T. \tag{54}$$

We consider a Policy Iteration setting, which can be described as the problem of improving the performance of an initially stabilizing controller. The initially stabilizing controller is obtained by selecting $\theta(0)$ to have the coefficients of the Lyapunov matrix that solves the corresponding linear quadratic regulator. We consider a Model-based Policy Iteration, in which the model is used to obtain the next state of the system.

In RLS-TD(λ) algorithm, we set the discount rate $\gamma = 0.95$, $\lambda = 0.8$, which are the same values suggested in Reference 1. In RLS-TD with forgetting factor algorithm, we set the discount rate $\gamma = 0.95$ (which is the same as RLS-TD(λ) algorithm), the weighting coefficient $a_k = 1$, the forgetting factor $\beta = 0.95$, and the normalizing signal $m_k^2 = 1 + \phi_t^T\phi_t$. In both algorithms, we set $P_0 = 10I$. Most of these values are found by trial-and-error experiments. We select the initial condition $x = 0; \dot{x} = 0; \vartheta = \pi + \pi/36 (i.e., 185°); \dot{\vartheta} = 0$: the learning process (evolution of the states and inputs) resulting from the two algorithms is sketched in Figures 3 and 4.

For the same initial condition, in order to check if the algorithms are effectively learning, we apply the controller offline. The term εofflineε refers to using the estimate obtained at time t in a new experiment with the same initial condition $x = 0; \dot{x} = 0; \vartheta = \pi + \pi/36 (i.e., 185°); \dot{\vartheta} = 0$. By doing this, it is easy one can calculate the corresponding cost to check if the algorithm is improving its performance or not. The results are shown in Figures 5 and 6, where it can be seen that the offline reward is improving, especially in the initial phase. The proposed RLS-TD with forgetting factor method seems to improve more as compared to the initial stabilizing controller. The learning is more effective when the system states are far from the equilibrium, which is when the regressor $\phi_t(s)$ assumes larger values: as the system approaches the

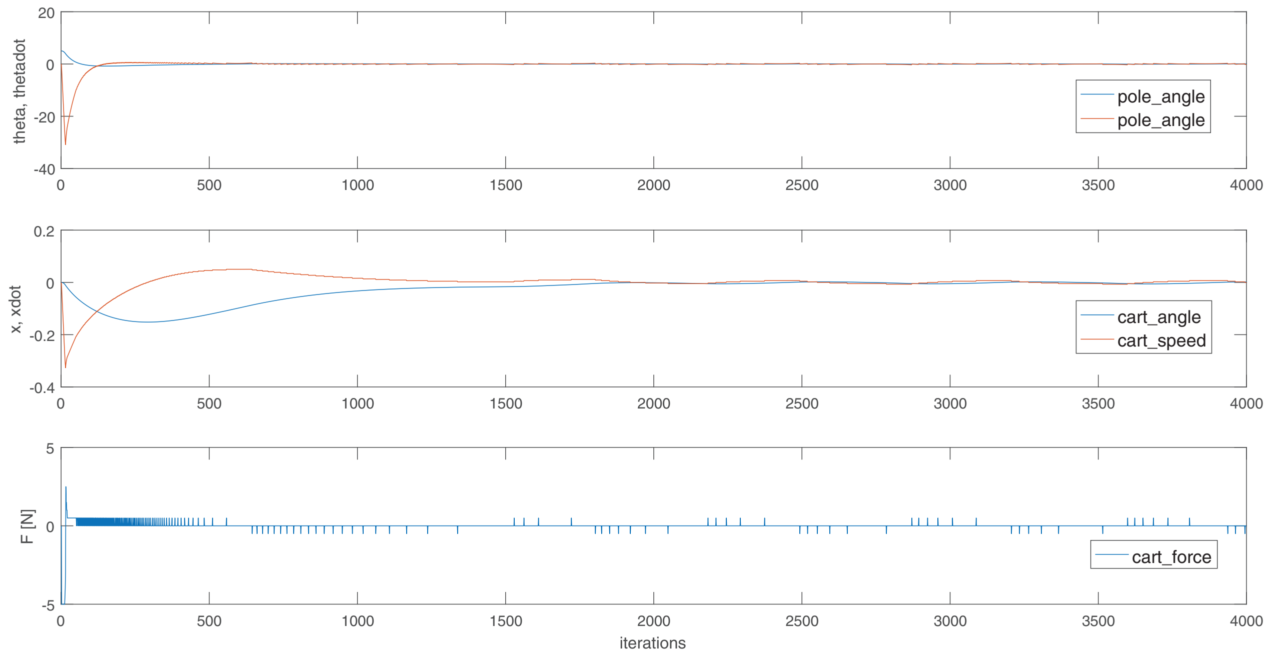


FIGURE 3 Cart/pole states and inputs using RLS-TD(λ)

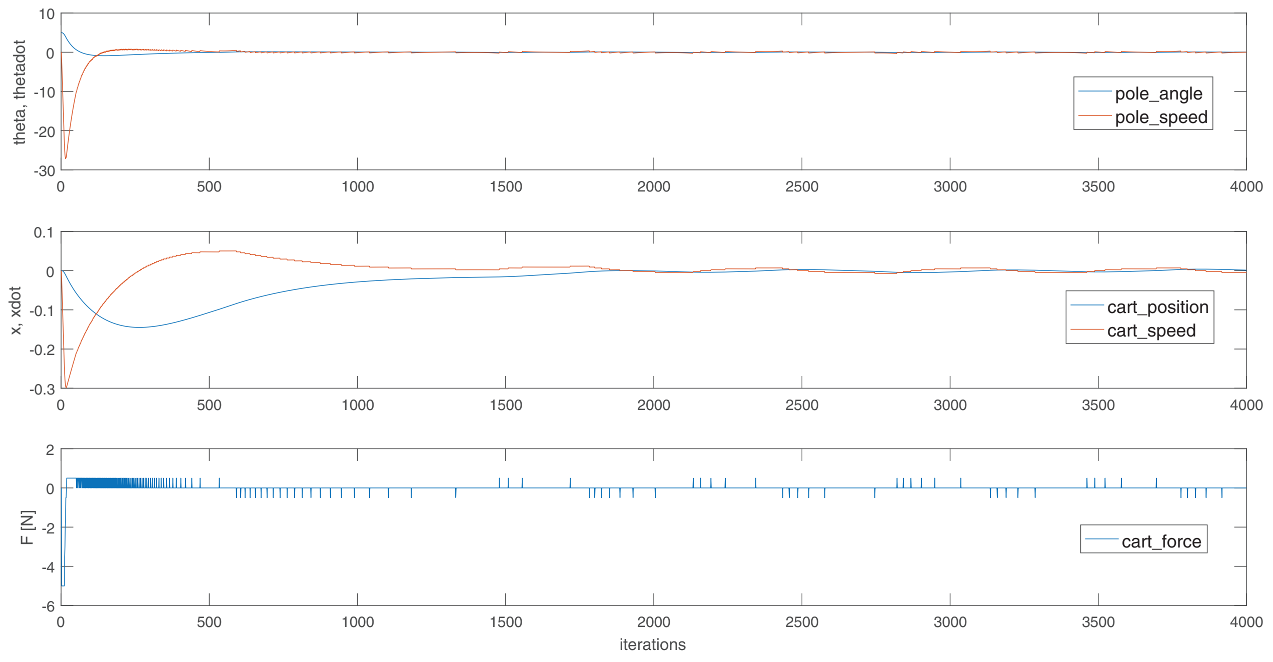
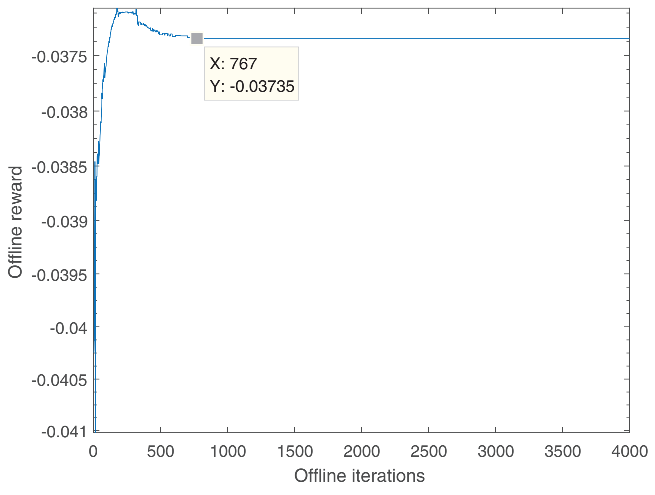


FIGURE 4 Cart/pole states and inputs using proposed RLS-TD-f

FIGURE 5 Offline reward of RLS-TD(λ)



equilibrium, the regressor $\phi_t(s)$ assumes smaller values and the learning becomes smaller. It can be seen that when the system has converged to the equilibrium, the learning curve might even become a bit worse. Dependence of learning on the quality of data is a well known phenomenon in learning schemes.[10]

In the offline simulation experiment, the reward range of RLS-TD(λ) algorithm starts from -0.04102 and reaches -0.03708 (best value): after that, it becomes a bit worse and converges to -0.03735. The reward range of the new algorithm starts from -0.04025 and reaches -0.03687 (best value): after that, it becomes a bit worse and converges to -0.03697. The evolution of the weights θ_t for both algorithms, are in Figures 7 and 8. One can see that, for both algorithms, a comparable trend in the evolution.

Some important information about the experimental results is shown in the Table 1. In the table, the integral norm is the integral from 0 to t (in line with the linear quadratic regulator theory), and $\chi = [x, \dot{x}, \vartheta, \dot{\vartheta}]^T$. It can be seen that the cost of the RLS-TD with forgetting factor algorithm is typically smaller. The third row of Table 1 refers to the best reward obtained by using the current parameters offline. The last row of Table 1 refers to the iteration at which the reward within $\pm 2\%$ of the final reward, that is, $-0.03735(1 \pm 2\%)$ and $-0.03697(1 \pm 2\%)$ This gives a measure of convergence speed.

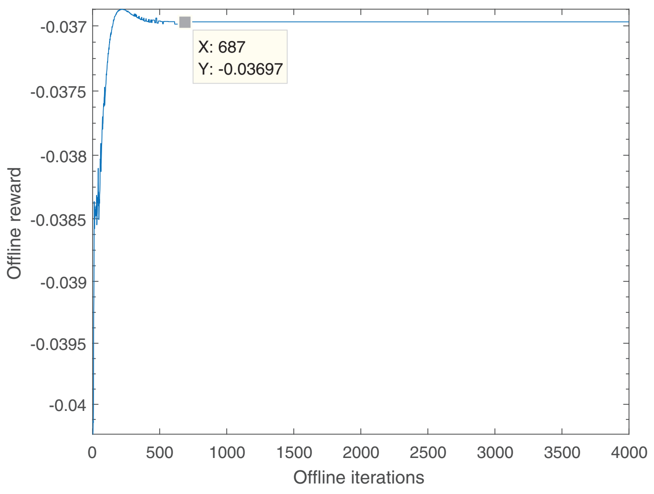


FIGURE 6 Offline reward of proposed RLS-TD-f

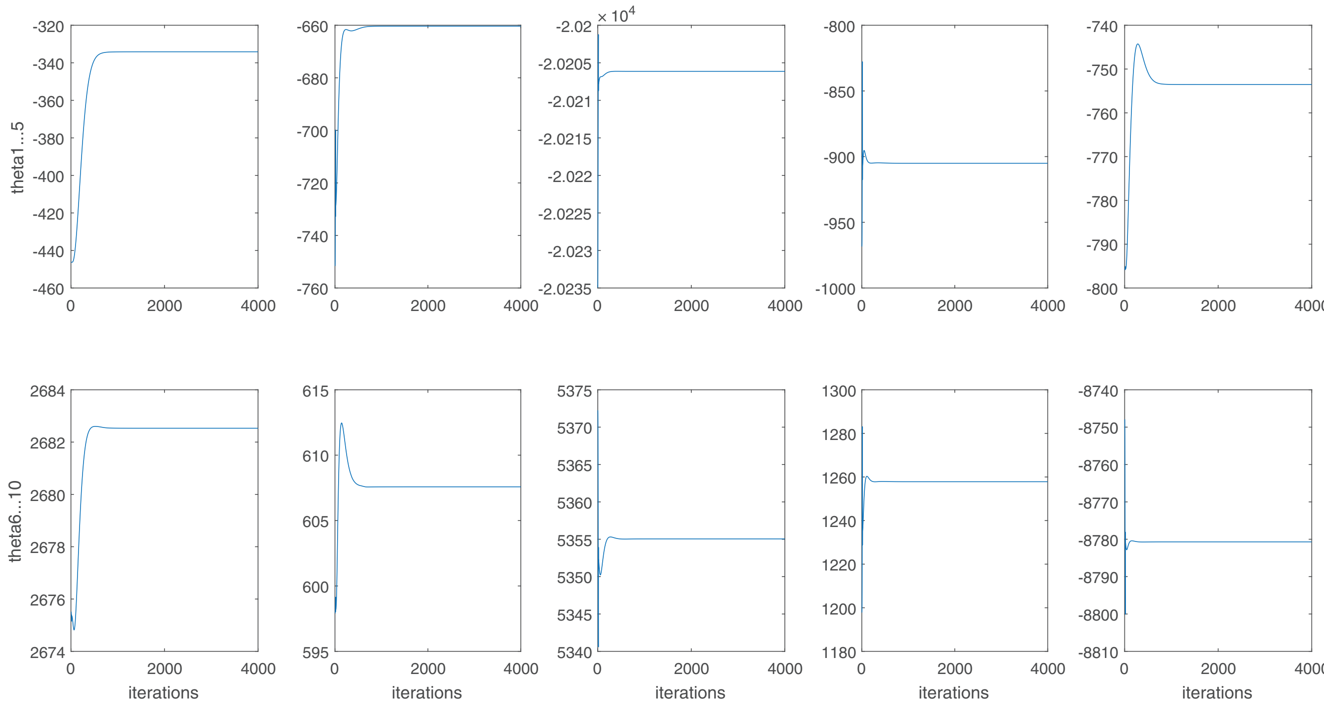


FIGURE 7 Weight parameter θ of RLS-TD(λ)

The previous experiments have been performed for initial condition $x = 0; \dot{x} = 0; \vartheta = \pi + \pi/36 (i.e., 185°); \dot{\vartheta} = 0$. In order to validate these benefits over a wider portion of the state space, we select four groups of initial conditions according to the following gridding method.

We select 9 values of initial conditions for the position x of the cart: $-2m, -1.5m, \dots, 1.5m, 2m$. We select 9 values of initial conditions for the angle ϑ of the pole (in degrees for better comprehension): $180-5°, 180-3.75°, \dots, 180+3.75°, 180+5°$. Similarly, we select 9 initial conditions for the speed $\dot{x}$ of the cart: $-1m/s, -0.75m/s, \dots, 0.75m/s, 1m/s$. Finally, we select 9 initial conditions for the angular speed $\dot{\vartheta}$ of the pole: $-10°/s, -7.5°/s, \dots, 7.5°/s, 10°/s$.

All these initial conditions crossed two by two ($9 \times 9 = 81$ initial conditions), for all possible 4 combinations, so as to obtain 4 groups of 81 initial conditions. To make the results more clear, Tables 2 and 3 report the final reward associated to the first group of 81 initial conditions. The tables for the other 3 groups are not reported for lack of space: however, in order to compare the approaches for all groups, we report in Table 4 how many times each algorithm overcomes the other one in terms of offline final reward and offline convergence. Table 4 shows that the proposed algorithm has a better

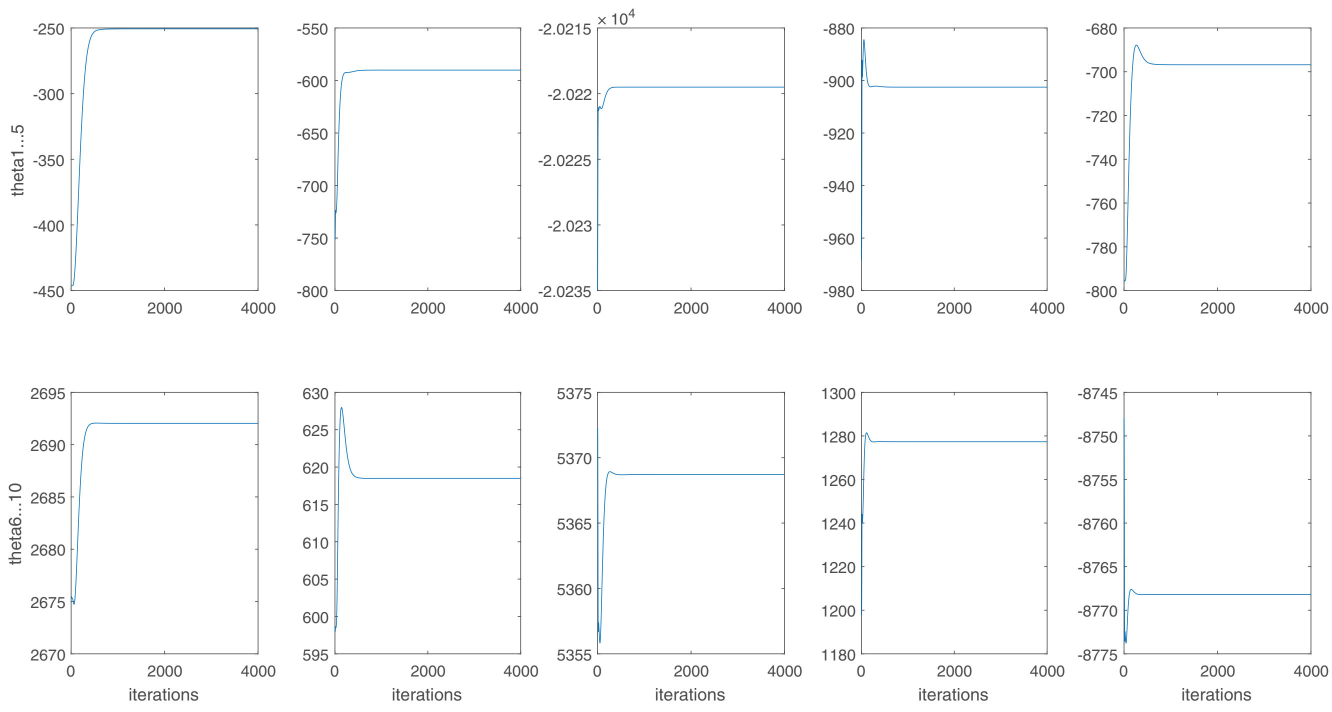


FIGURE 8 Weight parameter θ of proposed RLS-TD-f

TABLE 1 Comparison of experimental results

	RLS-TD(λ)	RLS-TD-f
Integral norm of $\chi^T\chi$	23.36	21.58
Integral norm of ρF^2	38.40	33.00
Largest offline reward	−0.0371	−0.0369
Convergence of offline learning	767	687

offline final reward in 70.4% of the cases, but in 67.6% of the cases more iterations are needed to converge to such better reward. Remarkably, the better reward is achieved while obtaining better online regulation and control effort (smaller norm of the state in 53.1% of the cases and smaller norm of the control input in 57.1% of the cases): this can partially explain the longer convergence time, since it is known in parameter estimation literature that smaller signals will make convergence slower.

5.2 | Adaptive cruise control (ACC)

Automatic cruise control (CC) and automatic adaptive cruise control (ACC) are excellent examples of a feedback control system. Nowadays, CC and ACC can be found in many modern vehicles. The purpose of CC is to maintain a constant speed despite external disturbances (wind, road grade, etc.). The purpose of ACC system is slightly more complex, in the sense that the speed may not be constant, but decided by a preceding vehicle with respect to which the ego vehicle wants to keep a desired distance. A CC protocol measures the vehicle speed, compares it to the desired speed, and automatically adjusts the throttle according to a control law. ACC also includes a radar to measure the relative position and relative velocity with respect to the preceding vehicle. Both the inverted pendulum and the cruise control system are benchmark examples often used as control test cases, for example see the Matlab tutorial https://ctms.engin.umich.edu/CTMS/index.php or the book.[33]

We consider here a simple model of the vehicle dynamics, compare Figure 9. It consists of a vehicle that can be controlled though a force or, more practically, through its acceleration (i.e., acceleration/brake pedal). From Newton's

TABLE 2 The first group of experimental results (reward)—RLS-TD(λ)

$x \setminus \vartheta$	−5	−3.75	−2.5	−1.25	0	1.25	2.5	3.75	5
−2	−0.8707	−0.8980	−0.9426	−1.0045	−1.0898	−1.1879	−1.2730	−1.3575	−1.5275
−1.5	−0.5162	−0.4966	−0.5037	−0.5300	−0.5643	−0.6073	−0.6563	−0.7102	−0.7966
−1	−0.2247	−0.2496	−0.2368	−0.2341	−0.2542	−0.2753	−0.2993	−0.3280	−0.3631
−0.5	−0.0571	−0.0577	−0.0626	−0.0617	−0.0687	−0.0817	−0.0979	−0.1159	−0.1483
0	−0.0374	−0.0238	−0.0123	−0.0039	0.0000	−0.0039	−0.0123	−0.0238	−0.0374
0.5	−0.1483	−0.1159	−0.0979	−0.0817	−0.0687	−0.0617	−0.0626	−0.0577	−0.0571
1	−0.3631	−0.3280	−0.2993	−0.2753	−0.2542	−0.2341	−0.2368	−0.2496	−0.2247
1.5	−0.7966	−0.7102	−0.6563	−0.6073	−0.5643	−0.5300	−0.5037	−0.4966	−0.5162
2	−1.5275	−1.3575	−1.2730	−1.1879	−1.0898	−1.0045	−0.9426	−0.8980	−0.8707

TABLE 3 The first group of experimental results (reward)—RLS-TD-f

$x \setminus \vartheta$	−5	−3.75	−2.5	−1.25	0	1.25	2.5	3.75	5
−2	−0.8581	−0.8809	−0.9114	−0.9489	−0.9881	−1.0299	−1.0756	−1.1269	−1.1859
−1.5	−0.4755	−0.4843	−0.5030	−0.5285	−0.5573	−0.5895	−0.6250	−0.6656	−0.7128
−1	−0.2264	−0.2086	−0.2171	−0.2320	−0.2519	−0.2742	−0.2994	−0.3293	−0.3643
−0.5	−0.0586	−0.0587	−0.0589	−0.0612	−0.0672	−0.0797	−0.0963	−0.1151	−0.1361
0	−0.0375	−0.0236	−0.0123	−0.0039	0.0000	−0.0039	−0.0123	−0.0236	−0.0375
0.5	−0.1361	−0.1151	−0.0963	−0.0797	−0.0672	−0.0612	−0.0589	−0.0587	−0.0586
1	−0.3643	−0.3293	−0.2994	−0.2742	−0.2519	−0.2320	−0.2171	−0.2086	−0.2264
1.5	−0.7128	−0.6656	−0.6250	−0.5895	−0.5573	−0.5285	−0.5030	−0.4843	−0.4755
2	−1.1859	−1.1269	−1.0756	−1.0299	−0.9880	−0.9489	−0.9114	−0.8809	−0.8581

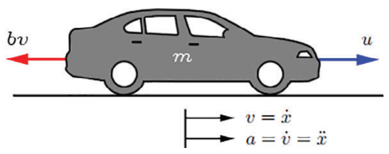


FIGURE 9 Model of Cruise Control

second law

$$m\dot{v} + bv = u \tag{55}$$

When this model is used for ACC as in Reference 33, it is assumed that both relative position and velocity can be measured. For this example, the parameters of the system are taken as in the Matlab tutorial https://ctms.engin.umich.edu/CTMS/index.php, shown in Table 5.

Similarly to the cart-pole, we consider a Policy Iteration setting, that is, improving the performance of an initially stabilizing controller. The initially stabilizing controller is obtained by a linear quadratic regulator. For the RLS-TD(λ) algorithm, we set discount rate $\gamma = 0.95$, $\lambda = 0.8$. In RLS-TD with forgetting factor algorithm, we set discount rate $\gamma = 0.95$, weighting coefficient $a_k = 1$, forgetting factor $\beta = 0.95$, and normalizing signal $m_k^2 = 0.5 + \phi_t^T \phi_t$. In both algorithms, we set $P_0 = I$.

Again, in order to check if the algorithms are effectively learning, we εofflineε apply the resulting estimate at time t and calculate the corresponding cost. We select a group of initial conditions corresponding to pairs of the two states of the system. The grid is composed of 64 points. The comparison results are summarized in Table 6.

TABLE 4 Comparison results for cart-pole system for each group of experiments

	Final reward		Convergence		Norm state		Norm input	
	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f
1st group	15/81 (18.5%)	**66**/81 (81.5%)	**62**/81 (76.5%)	19/81 (23.5%)	31/81 (38.3%)	**50**/81 (61.7%)	38/81 (46.9%)	**43**/81 (53.1%)
2nd group	30/81 (37.0%)	**51**/81 (63.0%)	**60**/81 (74.1%)	21/81 (25.9%)	**41**/81 (50.6%)	40/81 (49.4%)	**54**/81 (66.7%)	27/81 (33.3%)
3rd group	19/81 (23.5%)	**62**/81 (76.5%)	**58**/81 (71.6%)	23/81 (28.4%)	20/81 (24.7%)	**61**/81 (75.3%)	34/81 (42.0%)	**47**/81 (58.0%)
4th group	32/81 (39.5%)	**49**/81 (60.5%)	39/81 (48.1%)	**42**/81 (51.9%)	**60**/81 (74.1%)	21/81 (25.6%)	13/81 (16.0%)	**68**/81 (84.0%)
Total	96/324 (29.6%)	**228**/324 (70.4%)	**219**/324 (67.6%)	105/324 (32.4%)	152/324 (46.9%)	**172**/324 (53.1%)	139/324 (42.9%)	**185**/324 (57.1%)

TABLE 5 System parameters

m	vehicle mass	1000 kg
b	damping	50 Ns/m

TABLE 6 Comparison results for ACC for the group of experiments

	Final reward		Convergence		Norm state		Norm input	
	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f	RLS-TD(λ)	RLS-TD-f
CC	26/64 (40.6%)	**38**/64 (59.4%)	30/64 (46.9%)	**34**/64 (53.1%)	16/64 (25.0%)	**48**/64 (75.0%)	**54**/64 (84.4%)	10/64 (15.6%)
ACC	30/64 (46.9%)	**34**/64 (53.1%)	26/64 (40.6%)	**38**/64 (59.4%)	26/64 (40.6%)	**38**/64 (59.4%)	**48**/64 (75.0%)	16/64 (25.0%)

The table shows that the proposed algorithm has a better improvement of the initially stabilizing performance and faster convergence in 53.1–59.4% of the cases. The norm of the state is smaller in 59.4–75% of the cases, however the norm of the input is larger in 75.0–84.4%: this can partially explain the faster convergence time, since it is known in parameter estimation literature that larger signals will make convergence faster. In other words, the two benchmarks we used show that there is a trade-off between convergence time and high control input. However, the proposed approach can better improve the initially stabilizing control performance in both benchmarks.

6 | CONCLUSION

We have shown that the forgetting factor commonly used in least-squares algorithm has a similar role to the eligibility trace in temporal difference (TD) algorithm. We adopted an instrumental variable perspective and proposed a new algorithm. In least-square problems, the purpose of introducing forgetting factor is to give different weights to the data, so that the algorithm can respond to the changes of system to be controlled quickly. Because TD also has a gradient-descent structure, we have shown that forgetting factor can also be used in this setting. A further problem to be investigated analytically is the relationship between forgetting factor and eligibility trace. Currently, there is a qualitative relationship, which can be investigated in a quantitative way in the future. TD(λ) algorithm using eligibility trace has the characteristics of backward (considering the impact of past value on current value). Consequently, the qualitative similarity between forgetting factor and eligibility trace is that the influence of the past data on the current value is considered to some extent.

We have also used simulations to illustrate the method, performed in a policy iteration setting. Using extensive experiments on benchmark problems, we have shown that the proposed algorithm can better improve the initially stabilizing control performance.

ACKNOWLEDGMENTS

This work was partially supported by Jiangsu Double Innovation Talents Project for Jiangsu province grant 4207012004, and Natural Science Foundation of China grant 62073074.

ORCID

Simone Baldi https://orcid.org/0000-0001-9752-8925

REFERENCES

1. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press; 1998.
2. Bertsekas DP. *Reinforcement Learning and Optimal Control*. Belmont, MA: Athena Scientific; 2019.
3. Rishwaraj G, Ponnambalam SG, Loo CK. Heuristics-based trust estimation in multiagent systems using temporal difference learning. *IEEE Trans Cybern*. 2017;47(8):1925-1935.
4. Tesauro G. Practical issues in temporal difference learning. *Mach Learn*. 1992;8:257-277.
5. Keerthisinghe C, Verbič G, Chapman AC. A fast technique for smart home management: ADP with temporal difference learning. *IEEE Trans Smart Grid*. 2018;9(4):3291-3303.
6. Chen Z, Li L, Hu X, Yan B, Yang C. Temporal-difference learning-based stochastic energy management for plug-in hybrid electric buses. *IEEE Trans Intell Transp Syst*. 2019;20(6):2378-2388.
7. Sutton RS. Learning to predict by the methods of temporal differences. *Mach Learn*. 1988;3(1):9-44.
8. Tsitsiklis JN, Van Roy B. An analysis of temporal-difference learning with function approximation. *IEEE Trans Autom Control*. 1997;42(5):674-690.
9. Dayan P. The convergence of TD (λ) for general λ. *Mach Learn*. 1992;8(3-4):341-362.
10. Bradtke SJ, Barto AG. Linear least-squares algorithms for temporal difference learning. *Mach Learn*. 1996;22(1-3):33-57.
11. Young PC. *Recursive Estimation and Time-Series Analysis: An Introduction*. Berlin, Germany: Springer Science & Business Media; 2012.
12. Boyan JA. Technical update: least-squares temporal difference learning. *Mach Learn*. 2002;49(2-3):233-246.
13. Xu X, He H-g, Hu D. Efficient reinforcement learning using recursive least-squares methods. *J Artif Intell Res*. 2002;16:259-292.
14. Desouky SF, Schwartz HM. Q(λ)-learning adaptive fuzzy logic controllers for pursuit-evasion differential games. *Int J Adapt Control Signal Process*. 2011;25(10):910-927.
15. Kopf F, Ramsteiner S, Puccetti L, Flad M, Hohmann S. Adaptive dynamic programming for model-free tracking of trajectories with time-varying parameters. *Int J Adapt Control Signal Process*. 2020;34(7):839-856.
16. Dabney W, Thomas PS. Natural temporal difference learning. Paper presented at: Proceedings of the 28th AAAI Conference on Artificial Intelligence; 2014.
17. Blakeman S, Mareschal D. A complementary learning systems approach to temporal difference learning. *Neural Networks*. 2020;122:218–230.
18. Lian C, Xu X, Zuo L, Huang Z. Adaptive critic design with graph Laplacian for online learning control of nonlinear systems. *Int J Adapt Control Signal Process*. 2014;28(3-5):290-304.
19. Ramaswamy A, Bhatnagar S. Stability of stochastic approximations with "Controlled Markov" noise and temporal difference learning. *IEEE Trans Autom Control*. 2019;64(6):2614-2620.
20. Niu X, Xu J, Ren Q, Wang Q. Locomotion learning for an anguilliform robotic fish using central pattern generator approach. *IEEE Trans Ind Electron*. 2014;61(9):4780-4787.
21. Yeh K, Wu I, Hsueh C, Chang C, Liang C, Chiang H. Multistage temporal difference learning for 2048-like games. *IEEE Trans Comput Intell AI Games*. 2017;9(4):369-380.
22. Gao W, Jiang Z. Adaptive dynamic programming and adaptive optimal output regulation of linear systems. *IEEE Trans Autom Control*. 2016;61(12):4164-4169.
23. Bian T, Jiang ZP. Reinforcement learning and adaptive optimal control for continuous-time nonlinear systems: a value iteration approach. *IEEE Trans Neural Netw Learn Syst*. 2021;1-10.
24. Modares H, Lewis FL. Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning. *IEEE Trans Autom Control*. 2014;59(11):3051-3056.
25. Michailidis I, Baldi S, Kosmatopoulos EB, Ioannou PA. Adaptive optimal control for large-scale nonlinear systems. *IEEE Trans Autom Control*. 2017;62(11):5567-5577.
26. Pang B, Jiang Z-P, Mareels I. Reinforcement learning for adaptive optimal control of continuous-time linear periodic systems. *Automatica*. 2020;118:109035.
27. Modares H, Lewis FL. Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica*. 2014;50(7):1780-1792.

28. Khan SG, Herrmann G, Lewis FL, Pipe T, Melhuish C. Reinforcement learning and optimal adaptive control: an overview and implementation examples. *Annu Rev Control*. 2012;36(1):42-59.
29. Baldi S, Liu D, Zhang Z. On recursive temporal difference and eligibility traces. Paper presented at: Proceedings of the 46th Annual Conference of the IEEE Industrial Electronics Society IECON 2020; 2020:501-506. https://doi.org/10.1109/IECON43393.2020.9254411.
30. Devraj AM, Meyn SP. Differential TD learning for value function approximation. Paper presented at: Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC); 2016:6347-6354.
31. Singh S, Jaakkola T, Littman ML, Szepesvári C. Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach Learn*. 2000;38(3):287-308.
32. Yin B, Dridi M, El Moudni A. Recursive least-squares temporal difference learning for adaptive traffic signal control at intersection. *Neural Comput Appl*. 2019;31(2):1013-1028.
33. Ioannou P, Fidan B. *Adaptive Control Tutorial*. Philadelphia, PA: SIAM; 2006.
34. Kosmatopoulos EB. Adaptive control design based on adaptive optimization principles. *IEEE Trans Autom Control*. 2008;53(11):2680-2685.
35. Kosmatopoulos EB. An adaptive optimization scheme with satisfactory transient performance. *Automatica*. 2009;45(3):716-723.
36. Kosmatopoulos EB. Control of unknown nonlinear systems with efficient transient performance using concurrent exploitation and exploration. *IEEE Trans Neural Netw*. 2010;21(8):1245-1261.
37. Jha SK, Roy SB, Bhasin S. Initial excitation-based iterative algorithm for approximate optimal control of completely unknown LTI systems. *IEEE Trans Autom Control*. 2019;64(12):5230-5237.
38. Cho N, Shin H, Kim Y, Tsourdos A. Composite model reference adaptive control with parameter convergence under finite excitation. *IEEE Trans Autom Control*. 2018;63(3):811-818.

How to cite this article: Baldi S, Zhang Z, Liu D. Eligibility traces and forgetting factor in recursive least-squares-based temporal difference. *Int J Adapt Control Signal Process*. 2022;36(2):334-353. https://doi.org/10.1002/acs.3282