On recursive temporal difference and eligibility traces

Simone Baldi, Di Liu, Zichen Zhang

Abstract—This work studies a new reinforcement learning method in the framework of Recursive Least-Squares Temporal Difference (RLS-TD). Differently from the standard mechanism of eligibility traces, leading to RLS-TD(0), in this work we show that the forgetting factor commonly used in gradient-based estimation has a similar role to the mechanism of eligibility traces. We adopt an instrumental variable perspective to illustrate this point and we propose a new algorithm, namely — RLS-TD with forgetting factor (RLS-TD-f). We test the proposed algorithm in a Policy Iteration setting, i.e. when the performance of an initially stabilizing controller must be improved. We take the cart-pole benchmark as experimental platform; extensive experiments show that the proposed RLS-TD algorithm exhibits larger performance improvements in the largest portion of the state space.

Index Terms—Reinforcement learning; temporal difference; least-squares; instrumental variable method; eligibility traces.

I. INTRODUCTION

In recent years, reinforcement learning has been an active research area not only in machine learning [1], but also in control engineering [2], [3], and complex optimization and decision-making problems [4], [5], [6], [7]. Among the several reinforcement learning methods, temporal difference (TD) learning is one of the most popular one. The method dates back to 1988, when Sutton presented a method to estimate the value function of a Markov Decision Process (MDP) [8]: the method was named TD(λ) algorithm, where the parameter 0 < λ < 1 refers to the mechanism of eligibility traces. The TD(λ) was originally proposed for problems with finite state and action spaces. Since many real-world applications have large or infinite state space, value function approximation methods should be preferred. When combined with nonlinear value function approximators, TD(λ) can not guarantee convergence in general [9]. However, for TD(λ) with linear function approximators, also called linear TD(λ), Dayan showed convergence with 0 < λ < 1 [10].

Based on 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 [11]. The main difference between the least-squares and recursive least-squares version is whether the estimate of the value function is computed from a batch of data, or updated every time a new data arrives [12]. Later, Boyan proposed a class of linear temporal difference learning algorithms called LS-TD(λ) [13], where the mechanism of eligibility traces was embedded in the least-squares estimation: when RLS-TD(0) employs the mechanism of eligibility traces, the result is RLS-TD(λ), which was proposed and analyzed formally in [14].

The TD algorithm is the basic block of many reinforcement learning methods. For example, in [15], the author investigated the application of natural gradient descent to Bellman error based reinforcement learning algorithms. An algorithm known as Complementary Temporal Difference Learning (CTDL) was proposed in [16], which combines a Deep Neural Networks (DNN) with a Self-Organizing Map (SOM) updated by the TD error. See also [17], [18], [19] and references therein. Because TD is the basic block of many reinforcement learning methods, it is very important to study possible extensions and insights to the method. An open problem in the TD method regards the mechanism of eligibility traces. In fact, it is not clear how to select the parameter of the eligibility trace, and it is still an open question to give a clear interpretation to such mechanism [1]. In this work, we show that the mechanism of eligibility traces is analogous to the use of forgetting factor in parameter estimation methods.

With this observation, and considering the gradient structure of TD algorithm, we propose a new TD algorithm namely, RLS-TD with forgetting factor (RLS-TD-f). The effectiveness of the algorithm is tested in the cart-pole benchmark: it is shown that the proposed method overcomes RLS-TD(λ) over large portions of the state space.

The rest of the paper is organized as follows. In Sect. II, we introduce various elements and important concepts of reinforcement learning. Sect. III recalls TD(λ), LS-TD(λ) and RLS-TD(λ). In Sect. IV, we introduce the instrumental variable method to provide a new optimization-based perspective of RLS-TD. The new algorithm — RLS-TD with forgetting factor is proposed. In Sect. V, we test the new method on the cart-pole benchmark and compare it with RLS-TD(λ). Finally, Sect. VI contains concluding remarks.

II. BACKGROUND ON REINFORCEMENT LEARNING

Reinforcement learning problems involve learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. Essentially, a reinforcement learning problem describes a closed-loop problem because
the learning system’s actions influence its later inputs (cf. Fig. 1).

Fig. 1: The agent–environment interaction.

The value function is a crucial function in reinforcement learning since it specifies what is good in the long run.

\[
V_\pi(s) = \max_a E[r_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s, A_t = a]
\]

where \( r_{t+1}, r_{t+2}, \ldots \) is the sequence of rewards received after time step \( t \), \( \gamma \) is a parameter, \( 0 < \gamma \leq 1 \), called the discount rate, \( E[\cdot] \) denotes the expected value of a random variable given that the agent follows policy \( \pi \). 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. We call the function \( V_\pi \) the state-value function for policy \( \pi \).

Optimal value functions satisfy the Bellman equation

\[
V^*(s) = \max_a E[r_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a]
\]

where \( \max_a \) is the TD error, \( \alpha \) is the learning step size, \( \gamma \) is the reward discount rate.

The reason why researchers are interested in optimal value functions rather than optimal policies is that, once one has \( V^* \), it is relatively easy to determine an optimal policy via (2).

III. BACKGROUND ON TEMPORAL DIFFERENCE

The temporal difference (TD) method was initially proposed in [8] as a way to estimate the value function. The idea of TD is to calculate the value of the state through the value of the next state and form an iterative formula, known in literature as TD(0) formula

\[
V(s_t) \leftarrow V(s_t) + \alpha \delta_t
\]

where \( \delta_t = r_{t+1} + \gamma V(S_{t+1}) - V(S_t) \)

The reason why researchers are interested in optimal value functions rather than optimal policies is that, once one has \( V^* \), it is relatively easy to determine an optimal policy via (2).

The temporal difference (TD) method was initially proposed in [8] as a way to estimate the value function. The idea of TD is to calculate the value of the state through the value of the next state and form an iterative formula, known in literature as TD(0) formula

\[
V(s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s
\]

where \( r_{t+1}, r_{t+2}, \ldots \) is the sequence of rewards received after time step \( t \), \( \gamma \) is a parameter, \( 0 < \gamma \leq 1 \), called the discount rate, \( E[\cdot] \) denotes the expected value of a random variable given that the agent follows policy \( \pi \). 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. We call the function \( V_\pi \) the state-value function for policy \( \pi \).

Optimal value functions satisfy the Bellman equation

\[
V^*(s) = \max_a E[r_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a]
\]

where \( \max_a \) is the TD error, \( \alpha \) is the learning step size, \( \gamma \) is the reward discount rate.

The reason why researchers are interested in optimal value functions rather than optimal policies is that, once one has \( V^* \), it is relatively easy to determine an optimal policy via (2).

The reason why researchers are interested in optimal value functions rather than optimal policies is that, once one has \( V^* \), it is relatively easy to determine an optimal policy via (2).

TD methods can be viewed as gradient descent (hill climbing) in the space of the modifiable parameters (weights) [8]. That is, TD aims to minimize an error measure \( J(\theta) \)

\[
J(\theta_t) = \frac{1}{2} \sum_{k=1}^{t} |r_k - (\phi_k^T - \gamma \phi_k^{T+1} \theta_t) e_t|^2
\]

over the space of weights by repeatedly incrementing the weight vector \( \theta_t \) along the direction in which \( J(\theta) \) decreases most steeply. Denoting with \( \nabla_\theta J(\theta) \), the gradient of \( J(\theta) \) with respect to theta, (5) can be written as

\[
\theta_{t+1} = \theta_t + \alpha \Delta \theta_t, \quad \Delta \theta_t = -\alpha \nabla_\theta J(\theta_t).
\]

A. The eligibility trace mechanism

Eligibility traces are one of the basic mechanisms of reinforcement learning [21]. There is no clear interpretation of the eligibility trace mechanism: a popular interpretation [1] is that the eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. When a TD error occurs, only the eligible states or actions are assigned credit or blame for the error.

Temporal difference can be combined with eligibility trace, resulting in the TD(\( \lambda \)) algorithm:

\[
\theta_{t+1} = \theta_t + \alpha (r_{t+1} + \gamma \phi_{t+1}^T \theta_t - \phi_t^T \theta_t) e_t,
\]

\[
e_t = \gamma \lambda e_{t-1} + e_t
\]

where \( \lambda \) is a parameter that can be adjusted artificially, \( \lambda \in [0, 1] \). When \( \lambda = 1 \), the value change rate of each state in the past has a complete impact on \( \theta_{t+1} \), which can be regarded as Monte Carlo process; when \( \lambda = 0 \), the value change rate of each state in the past has no impact on \( \theta_{t+1} \), which can be regarded as TD(0) process; when \( \lambda \in (0, 1) \), each state in the past has an impact on adjustment \( \theta_{t+1} \) in a declining trend, the greater \( \lambda \), the greater the impact.

A convenient form of TD(\( \lambda \)) that exploits linear function approximator [9] is shown in Algorithm 1.

On each transition, the algorithm computes the scalar one-step TD error \( r_t + \gamma (\phi(s_t+1) - \phi(s_t))^T \theta_t \) and apportions that error among all state features according to their respective eligibilities \( e_t \).

B. Least-squares with eligibility trace

Boyan [13], pointed out some shortcomings of TD(\( \lambda \)) algorithm and put forward an improved method — Least-Squares TD(\( \lambda \)) (LS-TD(\( \lambda \))). As compared to TD(\( \lambda \)), LS-TD(\( \lambda \)) builds from experience matrix \( \sum_{k=1}^{t} (\phi_k^T - \gamma \phi_k^{T+1} \theta_t) e_t \), and a vector of dimension \( K \):

\[
b = \sum_{i=0}^{L} e_i r_i, \quad A = \sum_{i=0}^{L} e_i (\gamma \phi(s_{i+1}) - \phi(s_i))^T.
\]

After \( n \) independent trajectories have been observed, \( \theta_t \) can be estimated as \( A^{-1} b \). The complete LS-TD(\( \lambda \)) algorithm is specified in Algorithm 2.
Algorithm 1: TD(\(\lambda\)) algorithm:

Given: (1) a simulation model for a proper policy \(\pi\) in MDP \(S\);
(2) a featurizer \(\Phi : S \to \mathbb{R}^K\) mapping states to feature vectors, \(\Phi(\text{END}) \equiv 0\);
(3) a parameter \(\lambda \in [0, 1]\);
(4) learning step size: \(\alpha\);

Output: a coefficient vector \(\theta\) for which \(V_\lambda(s) \approx \phi^T(\theta)\).

Set \(\theta := 0\) (or an arbitrary initial estimate), \(t := 0\).

for \(n := 1, 2, \ldots\) do

Set \(\delta := 0\).

Choose a start state \(s_t \in S\).

while \(s_t \neq \text{END}\) do

Simulate one step of the process, producing a 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 \(e_{t+1} := \gamma \lambda e_t + \phi(s_{t+1})\).

Set \(t := t + 1\).

Set \(\theta := \theta + \alpha \delta e\).

endwhile

endfor

When \(\lambda = 0\), LS-TD(0) reduces to Bradtke and Barto’s LS-TD algorithm, which they derived using regression with instrumental variables [11].

C. Recursive least squares with eligibility trace

Least-squares methods require the computation of a matrix inverse at each time step, i.e. they have a computational complexity of \(O(K^3)\), assuming that the state representations are of length \(K\). Recursive Least-Squares (RLS) techniques have been used in [11] to derive a modified algorithm, RLS-TD(\(\lambda\)), with computational complexity of \(O(K^2)\).

\[\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma \phi_{t+1})^T P_{t-1} e_t} \delta_t e_t\] (10)

\[P_t = P_{t-1} - \frac{P_{t-1} e_t (\phi_t - \gamma \phi_{t+1})^T P_{t-1}}{1 + (\phi_t - \gamma \phi_{t+1})^T P_{t-1} e_t}\] (11)

\[\delta_t = r_t - (\phi_t - \gamma \phi_{t+1})^T \theta_{t-1}\] (12)

where the \(e_t\) is the eligibility trace, updated recursively according to

\[e_{t+1} = \gamma \lambda e_t + \phi(s_{t+1})\] (13)

It is recognized in literature that there are no clear rules to select \(\lambda\), and the value of \(\lambda\) should be selected appropriately to obtain the best approximation error [22].

IV. AN INSTRUMENTAL VARIABLE PERSPECTIVE OF RECURSIVE TEMPORAL DIFFERENCE

A. Gradient structure of TD algorithm

Going back to the standard TD method (5) for a while, [8] showed that the learning rule for TD(\(\lambda\)) (i.e. with eligibility trace mechanism) can also be written as a gradient update

\[\theta_{t+1} = \theta_t + \alpha [r_t + \gamma V_t(s_{t+1}) - V_t(s_t)] \sum_{k=1}^t \lambda^{t-k} \nabla_{\theta_t} V_t(s_k)\]

\[= \theta_t + \alpha \Delta \theta_t\] (14)

where

\[\Delta \theta_t = [r_t + \gamma V_t(s_{t+1}) - V_t(s_t)] \sum_{k=1}^t \lambda^{t-k} \nabla_{\theta_t} V_t(s_k)\]. (15)

It is worth noticing that the summation in term in (15) essentially keeps \(\Delta \theta_t\) fixed, whereas it goes through all the recently visited states \(s_k\). This is an attempt to separate the effects of changing the parameters from the effects of moving through the state space [11]. That is, (14) reveals a similar structure as the LS cost (6) where \(\theta_t\) is fixed and the data \(\omega_t\) change in the past [23]. The problem that we want to investigate in the following is to find the corresponding version of (6) for the update (14)-(15).

In order to explain this point, let us consider the goal of linear least-squares approximation of some function \(R : \mathbb{R}^n \to \mathbb{R}\), given samples of \(\omega_t \in \mathbb{R}^n\) and outputs \(r_t \in \mathbb{R}^n\). If only the observations \(r_t\) are corrupted by noise, then we have the following situation:

\[r_t = \omega_t^T \theta^* + \eta_t\] (16)

where \(\theta^*\) is the vector of true (but unknown) parameters and \(\eta_t\) is the output observation noise. Given the minimization of (6) (assuming \(\omega_t = \theta_t - \gamma \phi_{t+1}\)), it is well known that the optimal \(\theta_t\) can be found by setting the partial derivative
of \( J_t \) with respect to \( \theta_t \) equal to zero,
\[
\theta_t = \left( \sum_{k=1}^{t} \omega_k \omega_k^T \right)^{-1} \left( \sum_{k=1}^{t} \omega_k r_k \right) \tag{17}
\]

Relation (19) models the situation in which observation errors occur only on \( r_t \). If the input observations \( \omega_t \) are also noisy, the following errors-in-variables situation [12] applies:
\[
r_t = \hat{\omega}_t \theta^* - \zeta_t \theta^* + \eta_t \tag{18}
\]

Substituting \( \hat{\omega}_t \) directly for \( \omega_t \) in (17) has the effect of making the noise dependent upon the current state. This introduces a bias, and \( \theta_t \) no longer converges to \( \theta^* \) [12], [11]. One way around this problem is to introduce an instrumental variables \( \phi_t \), i.e. a vector that is correlated with the input \( \omega_t \), but uncorrelated with the observation noise \( \zeta_t \). The following equation is a modification of (17) that uses the instrumental variables and the noisy inputs:
\[
\theta_t = \left( \sum_{k=1}^{t} \phi_k \hat{\omega}_k \hat{\omega}_k^T \right)^{-1} \left( \sum_{k=1}^{t} \phi_k r_k \right) \tag{19}
\]

It was shown in [11] that LS-TD(0) can be interpreted using the instrumental variable method with \( \hat{\omega}_k = \phi_k - \gamma \phi_{k+1} \). In the following, we will use the instrumental variable method to provide a new perspective into the LS-TD(0), LSTD(\( \lambda \)) and their recursive versions.

B. Instrumental variable method as a minimizer

Let us first derive LS-TD(0) and RLS-TD(0) according to the instrumental variable method. This perspective is new as it provides an optimization cost for the RLS-TD(0) which was not available in previous TD literature. Let us consider the following minimizer in line with the LS-TD(0) in [11]
\[
\theta = (\Phi_k^T \Omega_k + P_0^{-1})^{-1} (P_0^{-1} \theta_0 + \Phi_k^T r_k) \tag{20}
\]
where \( \Phi_k = [\phi_1, \phi_2, ..., \phi_k] \) is the instrumental variable vector, \( \Omega_k = [\omega_1, \omega_2, ..., \omega_k] \) is the regressor vector, \( \hat{\omega}_k = \phi_k - \gamma \phi_{k+1} \), \( r_k = [r_1, r_2, ..., r_k] \) is the reward vector and \( \theta = [\theta_1, \theta_2, ..., \theta_k] \) is the weight vector. The parameters \( P_0 \) and \( \theta_0 \) will be defined later.

We want to find the cost corresponding to the instrumental variable minimizer (20). Because (20) is a minimizer, it must make the gradient of an appropriate cost function \( J \) equal to 0. From reverse calculation, we can get that the cost function
\[
J(\theta) = \frac{1}{2} (r_k - \Phi_k \theta)^T (r_k - \Phi_k \theta) - \frac{1}{2} (\Phi_k \theta)^T [\Omega_k - \hat{\omega}_k\theta] + \frac{1}{2} (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0). \tag{21}
\]

Instead of using (20) to calculate the minimizer, we want to do it in a recursive way [23] in order to update the minimizer when new data arrive,
\[
P_k = P_{k-1} - \frac{P_{k-1} \phi_k \omega_k^T P_{k-1}}{m_k^2 + \omega_k^T P_{k-1} \phi_k} \tag{22}
\]
\[
\theta_k = \theta_{k-1} + P_k \phi_k \frac{r_k - \theta_{k-1}^T \omega_k}{m_k^2} \tag{23}
\]
where \( m_k^2 \geq 1 \) is the normalizing signal designed to bound \( \phi_k \) from above.

C. Proposed method

Let us now consider the cost
\[
J(\theta) = \frac{1}{2} \sum_{k=1}^{t} a_k \beta^t-k \frac{r_k - \theta_k^T \phi_k}{m_k^2} - \frac{1}{2} \sum_{k=1}^{t} a_k \beta^t-k \frac{(\phi_k \hat{\theta}_k^T) [P_0^{-1} (\phi_k - \hat{\omega}_k)]}{m_k^2} + \frac{1}{2} \beta^t (\theta_k - \theta_0)^T P_0^{-1} (\theta_k - \theta_0) \tag{24}
\]

where \( a_k \) is a non-negative sequence of weighting coefficients, \( \beta \in (0, 1) \) is a design sequence.

Remark 1: Differently from (21), the cost (24) has a forgetting factor to strengthen the influence of current data and reduce the influence of historical data. This is essentially the same mechanism in (14)-(15). Therefore, cost (24) can be interpreted as the extension to (6) in the eligibility trace sense.

The algorithm for generating \( \theta_k \), the estimate of \( \theta^* \), is (derivation follows along similar lines as [23])
\[
P_k = \frac{1}{\beta} [P_{k-1} - \frac{a_k P_{k-1} \phi_k \hat{\omega}_k^T P_{k-1}}{m_k^2 \beta + a_k \omega_k^T P_{k-1} \phi_k}] \tag{25}
\]
\[
\theta_k = \theta_{k-1} + \sqrt{m_k} P_k \phi_k \frac{r_k - \theta_{k-1}^T \omega_k}{m_k^2} \tag{26}
\]

It is worth noticing the differences and similarities between the RLS-TD(\( \lambda \)) (10)-(13) and the proposed RLS-TD with forgetting factor (25)-(26). Such methods will now be compared experimentally.

V. VALIDATION

The balancing control of inverted pendulums is a typical nonlinear control problem, widely studied not only in control theory but also in artificial intelligence [1]. Fig. 2 shows the cart-pole balancing control system, which consists of a cart moving horizontally and a pole with one end fixed at the cart. The variable \( x \) is the horizontal position of the cart. The variable \( \vartheta \) is the angle of the pole with respect to the vertical (\( \vartheta = \pi \) is the upright position) and \( F \) is force applied to the cart. The dynamics of the system can be described by
\[
\ddot{x} = \frac{1}{m_c + m_p \sin^2 \vartheta} [F + m_p \sin \vartheta (\dot{\vartheta}^2 + g \cos \vartheta)] \tag{27}
\]

![Fig. 2: The cart-pole balancing control system](image-url)
\begin{equation}
\ddot{y} = \frac{1}{l(m_c + m_p \sin^2 \vartheta)} \times \\
\left[ -F \cos \vartheta - m_p l \dot{\vartheta}^2 \cos \vartheta \sin \vartheta - (m_c + m_p) g \sin \vartheta \right]
\end{equation}

where \(g\) is the acceleration due to the gravity, which is 9.81 m/s^2.

According to [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

\[ r = -x^2 - \dot{\vartheta}^2 - \dot{x}^2 - \dot{\vartheta}^2 - \rho F^2 \]

representing a quadratic cost. Where \(\rho\) is a design constant. In the simulation experiment, we set \(\rho = 0.1\). Furthermore, we set discount rate \(\lambda = 0.95\), \(\beta = 0.95\), and normalizing signal \(m_\varphi^2 = 1 + \Phi^T \Phi\). In both algorithms, we set \(P_0 = 10I\). The initial condition of the system are: \(x = 0; \dot{x} = 0; \dot{\vartheta} = \pi + \pi/36\) (i.e. 185°); \(\ddot{\vartheta} = 0\).

In RLS-TD(\(\lambda\)) algorithm, we set discount rate \(\gamma = 0.95\), \(\lambda = 0.8\), which are the standard values suggested in Sutton’s and Barto’s book. In RLS-TD with forgetting factor algorithm, we set discount rate \(\gamma = 0.95\) (as in Sutton’s and Barto’s book), weighting coefficient \(\alpha_0 = 1\), forgetting factor \(\beta = 0.95\), and normalizing signal \(m_\varphi^2 = 1 + \Phi^T \Phi\). In both algorithms, we set \(P_0 = 10I\). The initial condition of the system are: \(x = 0; \dot{x} = 0; \dot{\vartheta} = \pi + \pi/36\) (i.e. 185°); \(\ddot{\vartheta} = 0\).

Furthermore, in order to check if the algorithms are effectively learning, we “offline” apply the resulting estimate at time \(t\) and calculate the corresponding cost. This is represented in Fig. 3 and 4, where it can be seen that the offline reward is indeed improving. The proposed method seems to improve more as compared to the initial stabilizing controller. Also notice that the learning is more effective when the system states are far from the equilibrium: as the system approaches the equilibrium, the learning becomes smaller and might even become a bit worse. Dependence of learning on the quality of data is a well known phenomenon in learning schemes [11].

In the offline simulation experiment, the reward range of RLS-TD(\(\lambda\)) algorithm is \(-0.04102 \sim -0.03708\), converging to \(-0.03735\); the reward range of the new algorithm is \(-0.04025 \sim -0.03687\), converging to \(-0.03697\).

Some important information about the experimental results is shown in the Table I.

**TABLE I: Comparison of experimental results**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RLS-TD((\lambda))</th>
<th>RLS-TD-f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral norm of (\chi^T \chi) (state)</td>
<td>23.36</td>
<td>21.58</td>
</tr>
<tr>
<td>Integral norm of (\rho F^2) (input)</td>
<td>34.48</td>
<td>33.00</td>
</tr>
<tr>
<td>Largest offline learning reward</td>
<td>-0.0371</td>
<td>-0.0369</td>
</tr>
<tr>
<td>Steps needed for offline learning convergence</td>
<td>767</td>
<td>687</td>
</tr>
</tbody>
</table>

In the table, the integral norm is the integral from 0 to \(t\) (in line with the linear quadratic regulator theory), and

\[ \chi = [x, \dot{x}, \dot{\vartheta}, \dot{\vartheta}]^T. \]

It can be seen that the cost of the RLS-TD with forgetting factor algorithm is typically smaller.

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 combinations of initial conditions.

The set of values of initial position of the cart (i.e. \(x\)) is \(-2m, -1.5m, ..., 1.5m, 2m\). There are 9 values from \(-2m\) to \(2m\), and the step span is \(0.5m\). The set of values of initial angle of the pole from the vertical (i.e. \(\vartheta\)) is \(-5^\circ, -3.75^\circ, ..., 3.75^\circ, 5^\circ\). There are 9 values from \(-5^\circ\) to \(5^\circ\), and the step span is \(1.25^\circ\). The set of values of initial speed of the cart (i.e. \(\dot{x}\) ) is \(-1m/s, -0.75m/s, ..., 0.75m/s, 1m/s\). There are 9 values from \(-1m/s\) to \(1m/s\), and the step span is \(0.25m/s\). The set of values of initial angular speed of the pole (i.e. \(\dot{\vartheta}\) ) is \(-10^\circ/s, -7.5^\circ/s, ..., 7.5^\circ/s, 10^\circ/s\). There are 9 values from \(-10^\circ/s\) to \(10^\circ/s\), and the step span is \(2.5^\circ/s\).

All these initial conditions are combined into 4 groups of \(9 \times 9 = 81\) initial conditions. In order to compare the approaches, we count how many times each algorithm overcomes the other one in terms of offline final reward and offline convergence. The results are summarized in Table II. The table shows that the proposed algorithm has a better improvement of the initially stabilizing performance in 70.4% of the cases, but only in 32.4% of the cases, it achieves the convergence effect. 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.

<table>
<thead>
<tr>
<th>Final reward</th>
<th>Convergence</th>
<th>Norm state</th>
<th>Norm input</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLS-TD(λ)</td>
<td>RLS-TD(λ)</td>
<td>RLS-TD(λ)</td>
<td>RLS-TD(λ)</td>
</tr>
<tr>
<td>1st group</td>
<td>15/81</td>
<td>66/81</td>
<td>62/81</td>
</tr>
<tr>
<td></td>
<td>(18.5%)</td>
<td>(81.5%)</td>
<td>(76.5%)</td>
</tr>
<tr>
<td>2nd group</td>
<td>30/81</td>
<td>51/81</td>
<td>60/81</td>
</tr>
<tr>
<td></td>
<td>(37.0%)</td>
<td>(63.0%)</td>
<td>(74.1%)</td>
</tr>
<tr>
<td>3rd group</td>
<td>19/81</td>
<td>62/81</td>
<td>58/81</td>
</tr>
<tr>
<td></td>
<td>(23.5%)</td>
<td>(76.5%)</td>
<td>(71.6%)</td>
</tr>
<tr>
<td>4th group</td>
<td>32/81</td>
<td>49/81</td>
<td>39/81</td>
</tr>
<tr>
<td></td>
<td>(39.5%)</td>
<td>(60.5%)</td>
<td>(48.1%)</td>
</tr>
<tr>
<td>Total</td>
<td>96/324</td>
<td>230/324</td>
<td>219/324</td>
</tr>
<tr>
<td></td>
<td>(29.6%)</td>
<td>(70.4%)</td>
<td>(67.6%)</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this work we have shown that the forgetting factor commonly used in least-squares algorithm has a similar role to the eligibility trace in the TD algorithm. We adopted an instrumental variable perspective to illustrate this point and we proposed a new algorithm, namely — RLS-TD with forgetting factor. The purpose of introducing forgetting factor is to give different weights to the original data and the new data, so that the algorithm can respond to the changes of input process. Consequently, the similarity between forgetting factor and eligibility trace is that the influence of the past data on the current value is considered to some extent. Using extensive experiments on a cart-pole benchmark, we have shown that the proposed algorithm can better improve the initially stabilizing control performance: remarkably, this is done while achieving better online regulation and control effort.

REFERENCES