True Online TD(λ)

Harm van Seijen HARM.VANSEIJEN@UALBERTA.CA
Richard S. Sutton SUTTON@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, Canada

Abstract

TD(λ) is a core algorithm of modern reinforcement learning. Its appeal comes from its equivalence to a clear and conceptually simple forward view, and the fact that it can be implemented online in an inexpensive manner. However, the equivalence between TD(λ) and the forward view is exact only for the off-line version of the algorithm (in which updates are made only at the end of each episode). In the online version of TD(λ) (in which updates are made at each step, which generally performs better and is always used in applications) the match to the forward view is only approximate. In a sense this is unavoidable for the conventional forward view, as it itself presumes that the estimates are unchanging during an episode. In this paper we introduce a new forward view that takes into account the possibility of changing estimates and a new variant of TD(λ) that *exactly* achieves it. Our algorithm uses a new form of eligibility trace similar to but different from conventional accumulating and replacing traces. The overall computational complexity is the same as TD(λ), even when using function approximation. In our empirical comparisons, our algorithm outperformed TD(λ) in all of its variations. It seems, by adhering more truly to the original goal of TD(λ)—matching an intuitively clear forward view even in the online case—that we have found a new algorithm that simply improves on classical TD(λ).

1. Why True Online TD(λ) Matters

Temporal-difference (TD) learning is a core learning technique in modern reinforcement learning (Sutton, 1988; Kaelbling, Littman & Moore; Sutton & Barto, 1998; Szepesvári, 2014). One of the main challenges in reinforcement learning is to make predictions, in an initially unknown environment, about the (discounted) sum of future rewards, the *return*, based on currently observed feature and a certain behavior policy. With TD learning it is possible to learn good estimates of the expected return quickly by bootstrapping from other expected-return estimates. TD(λ) is a popular TD algorithm that combines basic TD learning with *eligibility traces* to further speed learning. The popularity of TD(λ) can be explained by its simple implementation, its low computational complexity, and its conceptually straightforward interpretation, given by its *forward view*.

The forward view of TD(λ) (Sutton & Barto, 1998) is that the estimate at each time step is moved toward an update target known as as the λ-*return*; the corresponding algorithm is known as the λ-*return algorithm*. The λ-return is an estimate of the expected return based on both subsequent rewards and the expected-return estimates at subsequent states, with λ determining the precise way these are combined. The forward view is useful primarily for understanding the algorithm theoretically and intuitively. It is not clear how the λ-return could form the basis for an online algorithm, in which estimates are updated on every step, because the λ-return is generally not known until the end of the episode. Much clearer is the off-line case, in which values are updated only at the end of an episode, summing the value corrections corresponding to all the time steps of the episode. For this case, the overall update of the λ-return algorithm has been proven equal to the off-line version of TD(λ), in which the updates are computed on each step, but not used to actually change the value estimates until the end of the episode, when they are summed to produce an overall update (Sutton & Barto, 1998). The overall per-episode updates of the λ-return algorithm and of off-line TD(λ) are the same, even though the updates on each time step are different. By the end of the episode they must sum to the same overall update.

One of TD(λ)'s most appealing features is that at each step it only requires temporally local information—the immediate next reward and next state; this enables the algorithm to be applied online. The online updates will be slightly dif-

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

ferent from those of the off-line version of TD(λ) because the estimates change during the episode and these participate in determining the λ-returns. Thus the overall effect of online TD(λ) by the end of the episode will not equal that of the off-line λ-return algorithm. They may be close if the step-size parameter is small and the episode is short, but in general the overall updates will differ. On balance this is a good thing (for online TD(λ)) because online TD updates are generally better than off-line ones (see Sutton & Barto, Sections 7.1–3). What is needed is not an online way to produce the same results as the (off-line) λ-return algorithm, but rather a new forward view for the online case, with a new online λ-return-like algorithm, and then a new version of TD(λ) that achieves the same result when applied online.

This is exactly what we provide in this paper. First we present a new online forward view, based on a modified λ-return, which we call the *truncated λ-return*. Then we present a new version of TD(λ) that obtains exact step-by-step equivalence with our online forward view. We call this variation *true online TD(λ)* because it is truer to the idea of TD(λ) as matching a λ-based forward view. Empirically, we demonstrate that true online TD(λ) also performs better on three benchmark problems.

2. Problem Setting and Notation

In this section, we formally present our learning framework, the various versions of TD(λ), and the conventional forward view. As a convention, we indicate random variables by capital letters (e.g., S_t, R_t), vectors by bold letters (e.g., $\boldsymbol{\theta}$, $\boldsymbol{\phi}$), functions by lowercase letters (e.g., v), and sets by calligraphic font (e.g., $\mathcal{S}$, $\mathcal{A}$).

2.1. Markov Reward Processes

We focus in this paper on discrete-time *Markov reward processes* (MRPs), which can be described as 4-tuples of the form $\langle \mathcal{S}, p, r, \gamma \rangle$, consisting of $\mathcal{S}$, the set of all states; $p(s'|s)$, the transition probability function, giving for each state $s \in \mathcal{S}$ the probability of a transition to state $s' \in \mathcal{S}$ at the next step; $r(s, s')$, the reward function, giving the expected reward after a transition from s to s'. γ is the discount factor, specifying how future rewards are weighted with respect to the immediate reward.

The *return* at time step t is the discounted sum of rewards observed after time step t:

$$G_t = \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} \,.$$

An MRP can contain *terminal states*, dividing the sequence of state transitions into *episodes*. When a terminal state is reached the current episode ends and the state is reset to the initial state. The return for an episodic MRP is defined as:

$$G_t = \sum_{i=1}^{T-t} \gamma^{i-1} R_{t+i} \,,$$

where T is the time step that the terminal state is reached.

We are interested in learning the value-function v of an MRP, which maps each state $s \in \mathcal{S}$ to the expected value of the return:

$$v(s) = \mathbb{E}\{G_t \,|\, S_t = s\} \,.$$

2.2. Function Approximation

In the general case, the state-space can be huge or even continuous. In such cases it is not possible to represent the value function v exactly. In this case, or when data-generalization is desired, function approximation is used to represent v. The approximate value-function $\hat{v}(s, \boldsymbol{\theta})$ gives the approximate value of state s, given a weight vector $\boldsymbol{\theta}$. Our learning algorithms produce a sequence of weight vectors $\boldsymbol{\theta}_t$, and as a shorthand we use $\hat{v}_t(s) = \hat{v}(s, \boldsymbol{\theta}_t)$.

A common approach is to use linear function approximation, in which case the value of a state is the inner product between the weight vector $\boldsymbol{\theta}$ and a state-dependent feature vector $\boldsymbol{\phi}(s)$. In this case, the value of state s at time step t is approximated by:

$$\hat{v}_t(s) = \boldsymbol{\theta}_t^\top \boldsymbol{\phi}(s) = \sum_i \theta_{i,t} \, \phi_i(s) \,,$$

where $\phi_i(s)$ is the value of feature i in state s, and $\theta_{i,t}$ is the weight of that feature at time step t. It s is a terminal state, then by definition $\boldsymbol{\phi}(s) = 0$, so that $\hat{v}_t(s) = 0$. We also use the shorthand $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$.

Although TD(λ) is not strictly speaking a gradient-descent method (Barnard, 1993) it is still useful to view it in these terms (as in Sutton & Barto, 1998). Stochastic gradient descent involves incremental updates of the weight vector $\boldsymbol{\theta}$ at each time step in the direction of the gradient of the time step's error. The updates can generally be written as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \, [U_t - \hat{v}_t(S_t)] \nabla_{\boldsymbol{\theta}_t} \hat{v}_t(S_t) \,, \qquad (1)$$

where U_t is the *update target* and $\nabla_{\boldsymbol{\theta}_t} \hat{v}_t(S_t)$ is the gradient of $\hat{v}$ with respect to $\boldsymbol{\theta}_t$. In the case of linear function approximation, this gradient has a particularly simple form:

$$\nabla_{\boldsymbol{\theta}_t} \hat{v}_t(S_t) = \boldsymbol{\phi}_t \,.$$

There are many ways to construct an update target U_t from observed states and rewards. For example, *Monte Carlo* methods use the full return as the update target:

$$U_t = G_t \,.$$

TD methods use an update target that is based on value estimates of other states. For example, the TD(0) update target is:

$$U_t = R_{t+1} + \gamma \hat{v}_t(S_{t+1}) \,. \tag{2}$$

The update (1) is an *online* update, meaning that the weight vector is updated at every time step t. Alternatively, the weight vector can be updated *off-line*. With off-line updating, the weight vector stays constant during an episode, and instead the weight corrections are collected on the side. Let the weight vector for (all of) episode k be $\boldsymbol{\theta}_k$. The weight correction for time step t of this episode is:

$$\boldsymbol{\Delta}_t = \alpha \left[U_t - \hat{v}_k(S_t) \right] \nabla_{\boldsymbol{\theta}_k} \hat{v}_k(S_t) \,.$$

After the episode has terminated, the weight vector is updated by adding all weight corrections collected during the episode:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \sum_{t=0}^{T-1} \boldsymbol{\Delta}_t \,.$$

Online updating not only has the advantage that it can be applied to non-episodic tasks, but it will in general also produce better value-function estimates under temporal-difference learning (see Sutton & Barto, 1998, Sections 7.1–3).

2.3. Linear TD(λ)

Under linear function approximation, the TD(0) method, based on update target (2), performs at time step $t+1$ the following update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \, \delta_t \boldsymbol{\phi}_t \,,$$

where

$$\delta_t = R_{t+1} + \gamma \, \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t \tag{3}$$

is called the *TD error*. This update only affects the weights θ_i for which ϕ_i is non-zero at time step t. The idea behind TD(λ) is to update weights for which ϕ_i was non-zero in the (near) past as well. This is implemented by means of an eligibility vector $\boldsymbol{e}$, which reflects how much each feature is 'eligible' for the current TD error. TD(λ) performs an update of each θ_i, with the current TD error, proportional to its trace value e_i:[1]

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta_t \boldsymbol{e}_t \,. \tag{4}$$

The original linear TD(λ) (Sutton, 1988) used *accumulating traces*, in which $\boldsymbol{e}_t$ is initialized to the zero vector, $\mathbf{0}$, and then updated according to:

$$\boldsymbol{e}_t = \gamma \lambda \boldsymbol{e}_{t-1} + \alpha \, \boldsymbol{\phi}_t \,, \tag{5}$$

where $\lambda \in [0, 1]$. Note that for $\lambda = 0$, the TD(λ) update reduces to the TD(0) update. Algorithm 1 shows pseudocode for linear TD(λ) with accummulating traces.

[1] We fold the step-size α into the eligibility vector.

Algorithm 1 linear TD(λ) with accummulating traces

```
initialize θ arbitrarily
loop {over episodes}
  initialize e = 0
  initialize S
  repeat {for each step in the episode}
    generate reward R and next state S' for S
    δ ← R + γ θ^⊤ φ(S') − θ^⊤ φ(S)
    e ← γλe + αφ(S)
    θ ← θ + δe
    S ← S'
  until S is terminal
end loop
```

Singh and Sutton (1996) introduced an interesting variation on the traces of TD(λ) called *replacing traces*. These traces do not increase with repeated visits to a state (or state feature) but are rather reset to a constant value (generally 1) each time the state (or state feature) is visited. The trace due to the re-visit does not add to the existing trace, it *replaces* it. Replacing traces result in faster and more reliable learning on many problems (Sutton, 1996). Replacing traces are usually defined only for discrete states or linear function approximation with binary features (that are either 1 or 0, present or not present). We generalize the update slightly to allow the features to take on any values, and include α in the trace, as follows:

$$e_{i,t} = \begin{cases} \gamma \lambda e_{i,t-1} & \text{if } \phi_{i,t} = 0 \\ \alpha \phi_{i,t} & \text{if } \phi_{i,t} \neq 0 \end{cases} \qquad \text{for all } i \,.$$

This update, with (4), is exactly the same as TD(λ) with replacing traces as it has been explored in the literature, and also allows us to use the same idea with features that take on non-binary values, as in the experiments later in this paper.

2.4. The Conventional Forward View

The conventional forward view relates TD(λ) (with accumulating traces) to the *λ-return algorithm*. This algorithm performs at each time step a standard update (1) with the *λ-return* as update target. The λ-return G_t^λ is an estimate of the expected return based on a combinations of rewards and other estimates:

$$G_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t,\boldsymbol{\theta}}^{(n)} \,,$$

with $\boldsymbol{\theta}$ the weight vector corresponding to the current episode, and $G_{t,\boldsymbol{\theta}}^{(n)}$ the n-step return, defined by:

$$G_{t,\boldsymbol{\theta}}^{(n)} = \sum_{i=1}^{n} \gamma^{i-1} R_{t+i} + \gamma^n \, \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t+n} \,. \tag{6}$$

For an episodic task, the λ-return is defined as:

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t,\boldsymbol{\theta}}^{(n)} + \lambda^{T-t-1}G_{t,\boldsymbol{\theta}}^{(T-t)}, \quad (7)$$

where T is the time step that the terminal state is reached. Because the value of a terminal state is always 0, the last n-step return in this equation is equal to the full return:

$$G_{t,\boldsymbol{\theta}}^{(T-t)} = \sum_{i=1}^{T-t}\gamma^{i-1}R_{t+i} = G_t\,.$$

Note that for $\lambda = 0$, G_t^λ is equal to the TD(0) update target (2). On the other hand, for $\lambda = 1$, G_t^λ is equal to the full return G_t.

It has been shown that off-line TD(λ) is equal to the (off-line) λ-return algorithm (Sutton & Barto, 1998; Sutton, 1988). However, online TD(λ) is only approximately equal to the online λ-return algorithm. In fact, no version of online TD(λ) can be constructed that matches the online λ-return algorithm exactly, because the λ-return uses rewards and states beyond the current time step. In Section 3, we present a new forward view that can be implemented online.

2.5. Other Variations on TD(λ)

Several variations on TD(λ) other than those treated in this paper have been suggested in the literature. Shapire and Warmuth (1996) introduced a variation of TD(λ) for which upper and lower bounds on the accuracy of the value estimate of the current state can be derived and proven. Maei, Sutton, and others (Maei, 2011; Maei & Sutton, 2010; Sutton et al, in preparation; Sutton et al., 2009) have explored generalizations of TD(λ)-like algorithms to *off-policy learning*, in which the behavior policy (generating the data) and the target policy (whose value function is being learned) are allowed to be different. Although in this paper we consider only the on-policy case, where these two policies must be the same, extending the ideas of true online TD(λ) to the off-policy case would be an interesting and natural avenue of future work.

3. An Online Forward View

The conventional forward view assumes that the value estimates are constant during an episode. Consequently, the online TD(λ) algorithm matches the forward view only approximately. In order to construct a version of online TD(λ) that matches the forward view exactly, we have to think more carefully about what we mean by the term 'online'. In Section 2.2 we specified that online learning means that the value estimates are updated at every time step. This definition does not rule out the possibility that the update at time step t, uses information beyond time step t. One could for example record a sequence of state transitions and rewards, and at the end of an episode use the full sequence to construct update targets for all visited states. When the λ-return is used as the basis for an online forward view, this broad interpretation of the term online is required, because the λ-return is an update target that is based on the full experience sequence.

On the other hand, if we look at online TD(λ) the term online seems to mean something more restrictive. Specifically, online TD(λ) uses for the update at time step t only information up to time step t. This restriction is what makes online TD(λ) very generally applicable. For example, it can be applied to non-episodic tasks, and can be easily generalized to control tasks. We will call the more restrictive definition of online, as applies to online TD(λ), *strict-online*. The goal of this section is to define a strict-online forward view.

To construct a strict-online forward view, we start by defining a version of the λ-return that is truncated at a specific time step. Let t' be the time step the λ-return is truncated. The *truncated λ-return* $G_t^{\lambda|t'}$ is defined as:

$$G_t^{\lambda|t'} = (1-\lambda)\sum_{n=1}^{t'-t-1}\lambda^{n-1}G_{t,\boldsymbol{\theta}_{t+n-1}}^{(n)} + \lambda^{t'-t-1}G_{t,\boldsymbol{\theta}_{t'-1}}^{(t'-t)}\,. \quad (8)$$

Note that this definition is closely related to the definition of regular λ-return for episodic tasks (7). Basically, T is replaced by t', and the weight vectors used in the n-step returns have now a specific time index.

The idea behind the new strict-online forward view is in essence a simple one: at each time step, the truncated λ-returns from all previous time steps are updated, such that they are now truncated at the current time step; the weight vector of the current time step is determined by sequentially performing TD backups, using the updated λ-returns, starting from the initial weight vector, $\boldsymbol{\theta}_{init}$. The weight vectors for the first three time steps are shown below. We use a second index for $\boldsymbol{\theta}$ to indicate at which time step the

λ-returns are truncated.[2]

$$\boldsymbol{\theta}_{0,0}: \quad \boldsymbol{\theta}_{0,0} = \boldsymbol{\theta}_{init}\,,$$

$$\begin{aligned}\boldsymbol{\theta}_{1,1}: \quad \boldsymbol{\theta}_{1,0} &= \boldsymbol{\theta}_{init} \\ \boldsymbol{\theta}_{1,1} &= \boldsymbol{\theta}_{1,0} + \alpha_0\Big[G_0^{\lambda|1} - \boldsymbol{\theta}_{1,0}^\top \boldsymbol{\phi}_0\Big]\boldsymbol{\phi}_0\,,\end{aligned}$$

$$\begin{aligned}\boldsymbol{\theta}_{2,2}: \quad \boldsymbol{\theta}_{2,0} &= \boldsymbol{\theta}_{init} \\ \boldsymbol{\theta}_{2,1} &= \boldsymbol{\theta}_{2,0} + \alpha_0\Big[G_0^{\lambda|2} - \boldsymbol{\theta}_{2,0}^\top \boldsymbol{\phi}_0\Big]\boldsymbol{\phi}_0\,, \\ \boldsymbol{\theta}_{2,2} &= \boldsymbol{\theta}_{2,1} + \alpha_1\Big[G_1^{\lambda|2} - \boldsymbol{\theta}_{2,1}^\top \boldsymbol{\phi}_1\Big]\boldsymbol{\phi}_1\,,\end{aligned}$$

$$\begin{aligned}\boldsymbol{\theta}_{3,3}: \quad \boldsymbol{\theta}_{3,0} &= \boldsymbol{\theta}_{init} \\ \boldsymbol{\theta}_{3,1} &= \boldsymbol{\theta}_{3,0} + \alpha_0\Big[G_0^{\lambda|3} - \boldsymbol{\theta}_{3,0}^\top \boldsymbol{\phi}_0\Big]\boldsymbol{\phi}_0\,, \\ \boldsymbol{\theta}_{3,2} &= \boldsymbol{\theta}_{3,1} + \alpha_1\Big[G_1^{\lambda|3} - \boldsymbol{\theta}_{3,1}^\top \boldsymbol{\phi}_1\Big]\boldsymbol{\phi}_1\,, \\ \boldsymbol{\theta}_{3,3} &= \boldsymbol{\theta}_{3,2} + \alpha_2\Big[G_2^{\lambda|3} - \boldsymbol{\theta}_{3,2}^\top \boldsymbol{\phi}_2\Big]\boldsymbol{\phi}_2\,,\end{aligned}$$

More generally, the weight vector $\boldsymbol{\theta}_{t,k}$ is incrementally defined by the equation:

$$\boldsymbol{\theta}_{t,k} = \boldsymbol{\theta}_{t,k-1} + \alpha_{k-1}\Big[G_{k-1}^{\lambda|t} - \boldsymbol{\theta}_{t,k-1}^\top \boldsymbol{\phi}_{k-1}\Big]\boldsymbol{\phi}_{k-1} \quad (9)$$

for $0 < k \leq t$, with $\boldsymbol{\theta}_{t,0} = \boldsymbol{\theta}_{init}$.

Note that $\boldsymbol{\theta}_{i,j}$ for $j < i$ are temporary 'helper' vectors. In the context of the true online forward view, $\boldsymbol{\theta}_t$ refers to $\boldsymbol{\theta}_{t,t}$. For example, the value estimate of state s at time step t is:

$$\hat{v}_t(s) = \boldsymbol{\theta}_t^\top \boldsymbol{\phi}(s) \;= \boldsymbol{\theta}_{t,t}^\top \boldsymbol{\phi}(s)\,.$$

We call the algorithm that computes $\boldsymbol{\theta}_{t,t}$ at each time step t the *truncated λ-return algorithm*. Note that for $\lambda = 0$ the following holds:

$$G_t^{0|t'} = G_{t,\boldsymbol{\theta}_t}^{(1)} = R_{t+1} + \gamma \boldsymbol{\theta}_t \boldsymbol{\phi}_{t+1}\,.$$

Hence, for $\lambda = 0$ the truncated λ-return algorithm is equivalent to TD(0). For $\lambda = 1$, at the end of an episode ($t' = T$) the following holds:

$$G_t^{1|T} = G_{t,\boldsymbol{\theta}_{T-1}}^{(T-t)} = G_t\,.$$

Hence, for $\lambda = 1$ the values at the end of an episode computed by the truncated λ-return algorithm are equal to those computed by (an online, step-size version of) Monte-Carlo.

The truncated λ-return algorithm is an expensive method, and requires storage of all observed states and rewards. In the next section, we introduce a new online TD(λ) version that implements the strict-online forward view efficiently using eligibility traces.

[2]For ease of exposition, we focus on the linear case. However, our approach can be extended to the non-linear case in a straightforward way.

4. True Online TD(λ)

This section present the online TD(λ) method that matches the new online forward view exactly. First the algorithm itself is presented, then its equivalence to the new online forward view is proven. The section finishes with empirical results demonstrating the benefits of the algorithm.

4.1. The Algorithm

True online TD(λ) forms the backward view of the truncated λ-return algorithm. Like traditional TD(λ), it updates feature weights proportional to a decaying eligibility trace.

The three update equations that specify the algorithm are as follows:

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}_t && (10)\\ \boldsymbol{e}_t &= \gamma\lambda \boldsymbol{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t - \alpha_t \gamma \lambda [\boldsymbol{e}_{t-1}^\top \boldsymbol{\phi}_t]\, \boldsymbol{\phi}_t && (11)\\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \delta_t\, \boldsymbol{e}_t + \alpha_t [\boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}_t - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t]\, \boldsymbol{\phi}_t && (12)\end{aligned}$$

Comparing these three equations with equations (3), (4) and (5) — the update equations for traditional TD(λ) — reveals that the the true online TD(λ) updates are different in the following ways:

- δ_t: $\boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}_t$ is used instead of $\boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t$.
- $\boldsymbol{e}_t$: the term $-\alpha_t \gamma \lambda [\boldsymbol{e}_{t-1}^\top \boldsymbol{\phi}_t]\, \boldsymbol{\phi}_t$ is added.
- $\boldsymbol{\theta}_{t+1}$: the term $\alpha_t [\boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}_t - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t]$ is added.

Algorithm 2 shows pseudo-code that implements the true online TD(λ) equations. For $\lambda = 0$, the algorithm is equivalent to (regular) TD(0).

Algorithm 2 true online TD(λ)

initialize $\boldsymbol{\theta}$ arbitrarily
loop {over episodes}
  initialize $\boldsymbol{e} = \mathbf{0}$
  initialize S
  $\hat{v}_S \leftarrow \boldsymbol{\theta}^\top \boldsymbol{\phi}(S)$
  repeat {for each step in the episode}
    generate reward R and next state S' for S
    $\hat{v}_{S'} \leftarrow \boldsymbol{\theta}^\top \boldsymbol{\phi}(S')$
    $\delta \leftarrow R + \gamma \hat{v}_{S'} - \hat{v}_S$
    $\boldsymbol{e} \leftarrow \gamma\lambda \boldsymbol{e} + \alpha\big[1 - \gamma\lambda\, \boldsymbol{e}^\top \boldsymbol{\phi}(S)\big]\, \boldsymbol{\phi}(S)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta \boldsymbol{e} + \alpha\big[\hat{v}_S - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S)\big]\, \boldsymbol{\phi}(S)$
    $\hat{v}_S \leftarrow \hat{v}_{S'}$
    $S \leftarrow S'$
  until S is terminal
end loop

4.2. Equivalence to the Online Forward View

In this section, we prove that the values computed by true online TD(λ) are exactly the same as those computed by the truncated λ-return algorithm. We start by deriving two lemmas.

Lemma 1 *$G_t^{\lambda|t'}$, used by the truncated λ-return algorithm, is related to δ_t, used by true online TD(λ) ((10)), as follows:*

$$G_t^{\lambda|t'+1} - G_t^{\lambda|t'} = (\gamma\lambda)^{t'-t}\delta_{t'} .$$

Proof The truncated λ-return algorithm uses $G_t^{\lambda|t'}$ ((8)) with $\boldsymbol{\theta}_{\tau(n)} = \boldsymbol{\theta}_{t+n-1}$, resulting in:

$$G_t^{\lambda|t'} = (1-\lambda)\sum_{n=1}^{t'-t-1}\lambda^{n-1}G^{(n)}_{t,\boldsymbol{\theta}_{t+n-1}} + \lambda^{t'-t-1}G^{(t'-t)}_{t,\boldsymbol{\theta}_{t'-1}} .$$

From this it follows that $G_t^{\lambda|t'+1}$ is:

$$G_t^{\lambda|t'+1} = (1-\lambda)\sum_{n=1}^{t'-t}\lambda^{n-1}G^{(n)}_{t,\boldsymbol{\theta}_{t+n-1}} + \lambda^{t'-t}G^{(t'+1-t)}_{t,\boldsymbol{\theta}_{t'}} .$$

Subtracting $G_t^{\lambda|t'}$ from $G_t^{\lambda|t'+1}$ results in

$$G_t^{\lambda|t'+1} - G_t^{\lambda|t'} = \lambda^{t'-t}[G^{(t'+1-t)}_{t,\boldsymbol{\theta}_{t'}} - G^{(t'-t)}_{t,\boldsymbol{\theta}_{t'-1}}] . \quad (13)$$

Similarly, using (6), it can be shown that subtracting $G^{(t'-t)}_{t,\boldsymbol{\theta}_{t'-1}}$ from $G^{(t'+1-t)}_{t,\boldsymbol{\theta}_{t'}}$ yields:

$$\begin{aligned} G^{(t'+1-t)}_{t,\boldsymbol{\theta}_{t'}} - G^{(t'-t)}_{t,\boldsymbol{\theta}_{t'-1}} &= \gamma^{t'-t}\big[R_{t'+1} + \gamma\boldsymbol{\theta}_{t'}^\top\boldsymbol{\phi}_{t'+1} - \\ &\quad \boldsymbol{\theta}_{t'-1}^\top\boldsymbol{\phi}_{t'}\big] \\ &= \gamma^{t'-t}\delta_{t'} . \end{aligned}$$

Substituting this result in (13) yields the lemma. ∎

Using Lemma 1, the following lemma can be derived.

Lemma 2 *The following equation holds:*

$$\boldsymbol{\theta}_{t+1,t} - \boldsymbol{\theta}_{t,t} = \gamma\lambda\delta_t\boldsymbol{e}_{t-1},$$

with δ_t and $\boldsymbol{e}_{t-1}$ defined by (10) and (11), respectively.

Proof The proof structure is a proof by induction. First we prove that if for some k, $1 \le k < t$, the following equation holds:

$$\boldsymbol{\theta}_{t+1,k} - \boldsymbol{\theta}_{t,k} = (\gamma\lambda)^{t+1-k}\delta_t\boldsymbol{e}_{k-1} , \quad (14)$$

then it also holds for $k+1$. We then show that it holds for $k = 1$. Hence, by induction, it also holds for $k = t$, proving the lemma.

Subtracting $\boldsymbol{\theta}_{t,k+1}$ from $\boldsymbol{\theta}_{t+1,k+1}$ using (9) yields:

$$\begin{aligned} \boldsymbol{\theta}_{t+1,k+1} - \boldsymbol{\theta}_{t,k+1} &= \boldsymbol{\theta}_{t+1,k} - \boldsymbol{\theta}_{t,k} \\ &\quad +\alpha_k\Big[G_k^{\lambda|t+1} - G_k^{\lambda|t} \\ &\quad -(\boldsymbol{\theta}_{t+1,k} - \boldsymbol{\theta}_{t,k})^\top\boldsymbol{\phi}_k\Big]\boldsymbol{\phi}_k \end{aligned}$$

If (14) holds for k, this can be rewritten as (using Lemma 1):

$$\begin{aligned} \boldsymbol{\theta}_{t+1,k+1} - \boldsymbol{\theta}_{t,k+1} &= (\gamma\lambda)^{t+1-k}\delta_t\boldsymbol{e}_{k-1} \\ &\quad +\alpha_k\Big[(\gamma\lambda)^{t-k}\delta_t \\ &\quad -(\gamma\lambda)^{t+1-k}\delta_t(\boldsymbol{e}_{k-1})^\top\boldsymbol{\phi}_k\Big]\boldsymbol{\phi}_k \\ &= (\gamma\lambda)^{t-k}\delta_t\Big[\gamma\lambda\boldsymbol{e}_{k-1} + \alpha_k\boldsymbol{\phi}_k \\ &\quad -\alpha_k\gamma\lambda(\boldsymbol{e}_{k-1}^\top\boldsymbol{\phi}_k)\boldsymbol{\phi}_k\Big] \\ &= (\gamma\lambda)^{t-k}\delta_t\boldsymbol{\theta}_k \end{aligned}$$

This proves that if (14) holds for k, it holds for $k+1$.

Computing $\boldsymbol{\theta}_{t+1,1} - \boldsymbol{\theta}_{t,1}$ using (9) yields:

$$\begin{aligned} \boldsymbol{\theta}_{t+1,1} - \boldsymbol{\theta}_{t,1} &= \boldsymbol{\theta}_{t+1,0} - \boldsymbol{\theta}_{t,0} \\ &\quad +\alpha_k\Big[G_0^{\lambda|t+1} - G_0^{\lambda|t} \\ &\quad -(\boldsymbol{\theta}_{t+1,0} - \boldsymbol{\theta}_{t,0})^\top\boldsymbol{\phi}_0\Big]\boldsymbol{\phi}_0 \end{aligned}$$

Using $\boldsymbol{\theta}_{t+1,0} = \boldsymbol{\theta}_{t,0} = \boldsymbol{\theta}_{init}$ and Lemma 1 this reduces to:

$$\boldsymbol{\theta}_{t+1,1} - \boldsymbol{\theta}_{t,1} = \alpha_k(\gamma\lambda)^t\delta_t\boldsymbol{\phi}_0$$

This demonstrates that (14) holds for $k = 1$.[3] By induction, it follows that (14) also holds for $k = t$, proving the lemma. ∎

We are now ready to prove our main theorem.

Theorem 1 *$\boldsymbol{\theta}_{t,t}$, computed by the truncated λ-return algorithm, is the same as $\boldsymbol{\theta}_t$, computed by true online TD(λ), for all time steps t.*

Proof Using (9), $\boldsymbol{\theta}_{t+1,t+1}$ can be written as:

$$\boldsymbol{\theta}_{t+1,t+1} = \boldsymbol{\theta}_{t+1,t} + \alpha_t\Big[G_t^{\lambda|t+1} - \boldsymbol{\theta}_{t+1,t}^\top\boldsymbol{\phi}_t\Big]\boldsymbol{\phi}_t . \quad (15)$$

Rewriting Lemma 2 as:

$$\boldsymbol{\theta}_{t+1,t} = \boldsymbol{\theta}_{t,t} + \gamma\lambda\delta_t\boldsymbol{e}_{t-1} ,$$

[3] Technically, initialization of $\boldsymbol{e}_t$ occurs for $t = -1$, that is, $\boldsymbol{e}_{-1} = \boldsymbol{0}$. Using (11) it follows that $\boldsymbol{e}_0 = \alpha_0\boldsymbol{\phi}_0$.

and substituting this in (15) yields:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1,t+1} &= \boldsymbol{\theta}_{t,t} + \gamma\lambda\delta_t \boldsymbol{e}_{t-1} + \alpha_t[R_{t+1} + \boldsymbol{\theta}_{t,t}^\top \boldsymbol{\phi}_{t+1}]\,\boldsymbol{\phi}_t \\
&\quad - \alpha_t[(\boldsymbol{\theta}_{t,t} + \gamma\lambda\delta_t \boldsymbol{e}_{t-1})^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t \\
&= \boldsymbol{\theta}_{t,t} + \gamma\lambda\delta_t \boldsymbol{e}_{t-1} + \alpha_t[\delta_t + \boldsymbol{\theta}_{t-1,t-1}^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t \\
&\quad - \alpha_t[\boldsymbol{\theta}_{t,t}^\top \boldsymbol{\phi}_t + \gamma\lambda\delta_t \boldsymbol{e}_{t-1}^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t \\
&= \boldsymbol{\theta}_{t,t} + \gamma\lambda\delta_t \boldsymbol{e}_{t-1} + \alpha_t\delta_t\,\boldsymbol{\phi}_t \\
&\quad - \alpha_t\gamma\lambda\delta_t[\boldsymbol{e}_{t-1}^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t \\
&\quad + \alpha_t[\boldsymbol{\theta}_{t-1,t-1}^\top \boldsymbol{\phi}_t - \boldsymbol{\theta}_{t,t}^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t \\
&= \boldsymbol{\theta}_{t,t} + \delta_t\,\boldsymbol{e}_t + \alpha_t[\boldsymbol{\theta}_{t-1,t-1}^\top \boldsymbol{\phi}_t - \boldsymbol{\theta}_{t,t}^\top \boldsymbol{\phi}_t]\,\boldsymbol{\phi}_t
\end{aligned}$$

This final equation shows that the weight update from one time step to the other for the truncated λ-return is the same as the weight update for true online TD(λ) ((12)). ∎

4.3. Empirical Results

We compare the performance of true online TD(λ) with traditional TD(λ) with accumulating and replacing traces on a random-walk task. The random-walk task is shown in Figure 1 for $N = 6$ (N being the total number of states, including the terminal state). In our experiment we use $N = 11$. The transition probability in the direction of the terminal state, p, is set to 0.9. Initial $\boldsymbol{\theta}$ is $\mathbf{0}$ and $\gamma = 0.99$.

We used linear function approximation with two types of features, resulting in two different tasks. The feature values of these two tasks are shown in Table 1 (for $N = 6$). In task 1, there are (at most) 3 non-zero features for each state. In task 2, the number of non-zero features is between 1 and $N - 1$. For the terminal state all features have value 0. The L^2-norm of $\boldsymbol{\phi}(s)$ is 1 for all non-terminal states.

For the considered features of both tasks, the true state values can be represented exactly with linear function approximation. We specify the performance of a method as the root-mean-squared (RMS) error of the value estimates of all non-terminal states at the end of an episode with respect to their true values (which are analytically determined), averaged over the first 10 episodes and 100 independent runs.

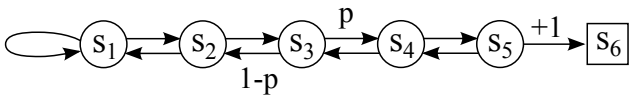

Figure 1. Random-walk for $N = 6$ states (experiments used $N = 11$). Transition probability to the right is p; transition probability to the left is $1 - p$. All rewards are 0, except transition to the terminal state (s_6), which results in a reward of +1. The initial state is s_1.

Figure 2 shows the performance on both tasks for α from 0 to 1.5 with steps of 0.01 and λ from 0 to 0.9 with steps of 0.1 and from 0.9 to 1.0 with steps of 0.025. Figure 3 shows the performance for the different λ values at the best α value.

Table 1. Feature values for random-walk task 1 and task 2 for $N = 6$.

		s_1	s_2	s_3	s_4	s_5	s_6
Task 1	ϕ_1	1	$1/\sqrt{2}$	$1/\sqrt{3}$	0	0	0
	ϕ_2	0	$1/\sqrt{2}$	$1/\sqrt{3}$	$1/\sqrt{3}$	0	0
	ϕ_3	0	0	$1/\sqrt{3}$	$1/\sqrt{3}$	$1/\sqrt{3}$	0
	ϕ_4	0	0	0	$1/\sqrt{3}$	$1/\sqrt{3}$	0
	ϕ_5	0	0	0	0	$1/\sqrt{3}$	0
Task 2	ϕ_1	1	$1/\sqrt{2}$	$1/\sqrt{3}$	$1/\sqrt{4}$	$1/\sqrt{5}$	0
	ϕ_2	0	$1/\sqrt{2}$	$1/\sqrt{3}$	$1/\sqrt{4}$	$1/\sqrt{5}$	0
	ϕ_3	0	0	$1/\sqrt{3}$	$1/\sqrt{4}$	$1/\sqrt{5}$	0
	ϕ_4	0	0	0	$1/\sqrt{4}$	$1/\sqrt{5}$	0
	ϕ_5	0	0	0	0	$1/\sqrt{5}$	0

While the return has an upper bound of 1, accumulating traces get errors above 1 on both tasks, indicating divergence of values. While replacing traces does not result in divergence of values for the considered step-sizes, it has no effect in the second task. True online TD(λ) is the only method that achieves a performance benefit (with respect to TD(0)) on both tasks.

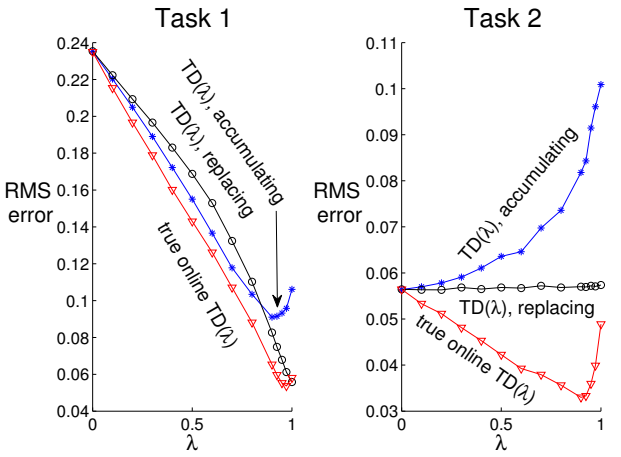

Figure 3. RMS error of state values at the end of each episode, averaged over the first 10 episodes, as well as 100 independent runs, for different values of λ at the best value of α.

5. Control

The true online TD(λ) algorithm (Algorithm 2) can be easily modified for control. Simply using a feature vector consisting of state-action features (i.e., using $\boldsymbol{\phi}(s, a)$ instead of $\boldsymbol{\phi}(s)$) changes the algorithm to a true online Sarsa(λ) algorithm.

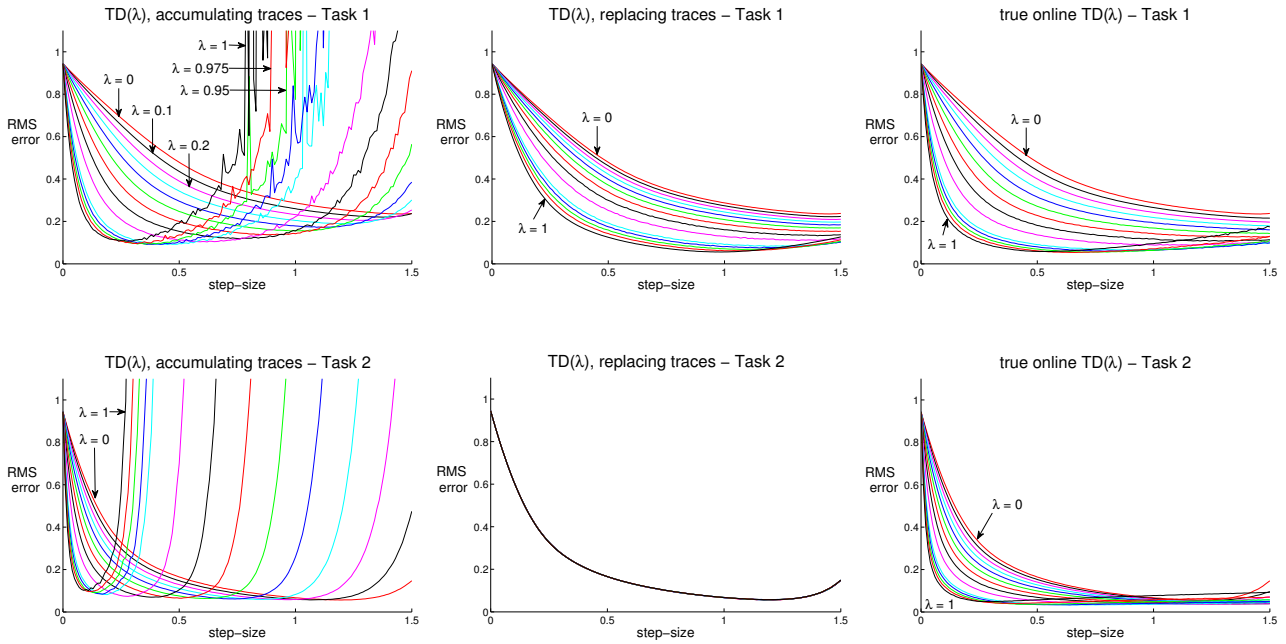

Figure 2. RMS error of state values at the end of each episode, averaged over the first 10 episodes, as well as 100 independent runs, for different values of α and λ.

Figure 4 compares true online Sarsa(λ) with the traditional Sarsa(λ) implementation on the standard mountain car task (Sutton & Barto, 1998) using 10 tilings of each 10×10 tiles. Results are plotted for $\lambda = 0.9$ and $\alpha = \alpha_0/10$, for α_0 from 0.2 to 2.0 with steps of 0.2. Clearing/no clearing refers to whether the trace values of non-selected actions are set to 0 (clearing) or not (no clearing), in case of replacing traces. The results suggest that the true online principle is also effective in a control setting.

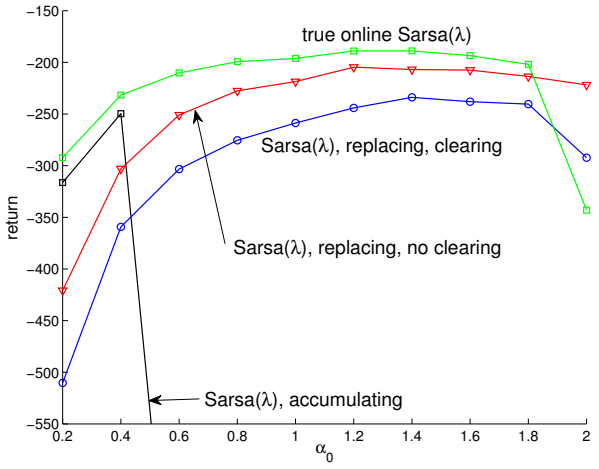

Figure 4. Average return over first 20 episodes on mountain car task for $\lambda = 0.9$ and different α_0. Results are averaged over 100 independent runs.

6. Conclusion

We presented for the first time an online version of the forward view which forms the theoretical and intuitive foundation for the TD(λ) algorithm. In addition, we have presented a new variant of TD(λ), with the same computational complexity as the classical algorithm, which we call true online TD(λ). We proved that true online TD(λ) matches the new online forward view exactly, in contrast to classical online TD(λ), which only approximates its forward view. In addition, we demonstrated empirically that true online TD(λ) outperforms conventional TD(λ) on three benchmark problems. It seems, by adhering more truly to the original goal of TD(λ)—matching an intuitively clear forward view even in the online case—that we have found a new algorithm that simply improves on TD(λ).

Acknowledgements

The authors thank Hado van Hasselt and Rupam Mahmood for extensive discussions leading to the refinement of these ideas. This work was supported by grants from Alberta Innovates – Technology Futures and the National Science and Engineering Research Council of Canada.

References

Barnard, E. (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics 23*(2):357–365.

Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research 4*:237–285.

Maei, H., Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In: *Proceedings of the Third Conference on Artificial General Intelligence*, pp. 91–96.

Schapire, R. E., and Warmuth, M. K. (1996). On the worst-case analysis of temporal-difference learning algorithms. *Machine Learning 22*(1/2/3):95–121.

Singh, S. P., and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning 22*(1/2/3):123–158.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning 3*(1):9–44.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts.

Sutton, R. S., Maei, H. R, Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs. and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 993–1000.

Sutton, R. S., Mahmood, A. R., Precup, D. (in preparation). A new Q(λ).

Szepesvári, Cs. (2010). *Algorithms for Reinforcement Learning*. Morgan and Claypool Press.