Investigating Practical Linear Temporal Difference Learning

Adam White
Department of Computer Science
Indiana University
Bloomington, IN 47405, USA
adamw@indiana.edu

Martha White
Department of Computer Science
Indiana University
Bloomington, IN 47405, USA
martha@indiana.edu

ABSTRACT

Off-policy reinforcement learning has many applications including: learning from demonstration, learning multiple goal seeking policies in parallel, and representing predictive knowledge. Recently there has been an proliferation of new policy-evaluation algorithms that fill a longstanding algorithmic void in reinforcement learning: combining robustness to off-policy sampling, function approximation, linear complexity, and temporal difference (TD) updates. This paper contains two main contributions. First, we derive two new hybrid TD policy-evaluation algorithms, which fill a gap in this collection of algorithms. Second, we perform an empirical comparison to elicit which of these new linear TD methods should be preferred in different situations, and make concrete suggestions about practical use.

Keywords

Reinforcement learning; temporal difference learning; off-policy learning

1. INTRODUCTION

Until recently, using temporal difference (TD) methods to approximate a value function from off-policy samples was potentially unstable without resorting to quadratic (in the number of features) computation and storage, even in the case of linear approximations. Off-policy learning involves learning an estimate of total future reward that we would expect to observe if the agent followed some target policy, while learning from samples generated by a different behavior policy. This off-policy, policy-evaluation problem, when combined with a policy improvement step, can can be used to model many different learning scenarios, such as learning from many policies in parallel [18], learning from demonstrations [1], learning from batch data [8], or simply learning about the optimal policy while following an exploratory policy, as in the case of Q-learning [25]. In this paper, we focus exclusively on the off-policy, policy evaluation problem, commonly referred to as value function approximation or simply the *prediction problem*. Over the past decade there has been an proliferation of new linear-complexity, policy-evaluation methods designed to be convergent in the off-policy case.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

These novel algorithmic contributions have focused on different ways of achieving stable off-policy prediction learning. The first such methods were the gradient TD family of algorithms that perform approximate stochastic gradient descent on the mean squared projected Bellman error (MSPBE). The primary drawback of these methods is the requirement for a second set of learned weights, a second step size parameter, and potentially high variance updates due to importance sampling. Empirically the results have been mixed, with some results indicating that TD can be superior in on-policy settings[17], and others concluding the exact opposite [4].

Later, provisional TD (PTD) was introduced [20] to rectify the issue that the bootstrap parameter λ, used in gradient TD methods[11] does not correspond well with the same parameter used by conventional TD learning [19]. Specifically, for $\lambda = 1$, gradient TD methods do not correspond to any known variant of off-policy Monte Carlo. The PTD algorithm fixes this issue, and in on-policy prediction is exactly equivalent to the conventional TD algorithm. PTD does not use gradient corrections, and is only guaranteed to converge in the tabular off-policy prediction setting. Its empirical performance relative to TD and gradient TD, however, is completely unknown.

Recently Sutton et al. [21] observed that conventional TD does not correct its update based on the notion of a follow-on distribution. This distributional mis-match provides another way to understand the off-policy divergence of conventional off-policy TD. They derive the Emphatic TD (ETD) algorithm that surprisingly achieves convergence [27] without the need for a second set of weights, like those used by gradient TD methods. Like gradient TD methods, however, it seems that this algorithm also suffers from high variance due to importance sampling. Hallak et al. [7] introduced a variant ETD that utilizes a scaling parameter β, which is meant to reduce the magnitude of the follow-on trace. Comparative empirical studies for ETD and ETD(β), however, have been limited.

The most recent contribution to this line of work explores a mirror-prox approach to minimizing the MSPBE [12, 13, 9]. The main benefit of this work was that it enabled the first finite sample analysis of an off-policy TD-based method with function approximation, and the application of advanced stochastic gradient optimizations. Liu et al. [9] introduced two mirror-prox TD algorithms, one based on the GTD2 algorithm [17] the other based on TDC [17][1] and showed

[1]The GTD2 and TDC algorithms are gradient TD methods that do not use eligibility traces; $\lambda = 0$.

that these methods outperform their base counter-parts on Baird's counterexample [2], but did not extend their new methods with eligibility traces.

A less widely known approach to the off-policy prediction problem is based on algorithms that perform precisely TD updates when the data is sampled on-policy, and corrected gradient-TD style updates when the data is generated off-policy. The idea is to exploit the supposed superior efficiency of TD in on-policy learning, while maintaining robustness in the off-policy case. These "hybrid" TD methods were introduced for state value-function based prediction [11], and state-action value-function based prediction [6], but have not been extended to utilize eligibility traces, nor compared with the recent developments in linear off-policy TD learning (many developed since 2014).

Meanwhile a separate but related thread of algorithmic development has sought to improve the operation of eligibility traces used in both on- and off-policy TD algorithms. This direction is based on another nonequivalence observation: the update performed by the forward view variant of the conventional TD is only equivalent to its backward view update at the end of sampling trajectories. The proposed true-online TD (TO-TD) prediction algorithm [24], and true-online GTD (TO-GTD) prediction algorithm [23] remedy this issue, and have been shown to outperform conventional TD and gradient TD methods respectively on chain domains. The TO-TD algorithm requires only a modest increase in computational complexity over TD, however, the TO-GTD algorithm is significantly more complex to implement and requires three eligibility traces compared to GTD. Nevertheless, both TO-TD and TO-GTD achieve linear complexity, and can be implemented in a completely incremental way.

Although there asymptotic convergence properties of many of these methods has been rigorously characterized, but empirically there is still much we do not understand about this now large collection of methods. A frequent criticism of gradient TD methods, for example, is that they are hard to tune and not well-understood empirically. It is somewhat disappointing that perhaps the most famous application of reinforcement learning—learning to play Atari games [15]—uses potentially divergent off-policy Q-learning. In addition, we have very little understanding of how these methods compare in terms of learning speed, robustness, and parameter sensitivity. By clarifying some of the empirical properties of these algorithms, we hope to promote more wide-spread adoption of these theoretically sound and computationally efficient algorithms.

This paper has two primary contributions. First, we introduce a novel extension of hybrid methods to eligibility traces resulting in two new algorithms, HTD(λ) and true-online HTD(λ). Second, we provide an empirical study of TD-based prediction learning with linear function approximation. The conclusions of our experiments are surprisingly clear:

1. GTD(λ) and TO-GTD(λ) should be preferred if robustness to off-policy sampling is required

2. Between the two GTD(λ) should be preferred if computation time is at a premium

3. Otherwise, TO-ETD(λ, β) was clearly the best across our experiments except on Baird's counterexample.

2. BACKGROUND

This paper investigates the problem of estimating the discounted sum of future rewards *online* and with function approximation. In the context of reinforcement learning we take online to mean that the agent makes decisions, the environment produces outcomes, and the agent updates its parameters in a continual, real-time interaction stream. We model the agent's interaction as Markov decision process defined by a countably infinite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, and a scalar discount function $\gamma : \mathcal{S} \to \mathbb{R}$. The agent's observation of the current situation is summarized by the feature vector $\mathbf{x}(S_t) \in \mathbb{R}^d$, where $S_t \in \mathcal{S}$ is the current state and $d \ll |\mathcal{S}|$. On each time step t, the agent selects an action according to it's *behavior policy* $A_t \sim \mu(S_t, \cdot)$, where $\mu : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The environment then transitions into a new state $S_{t+1} \sim P(S_t, A_t, \cdot)$, and emits a scalar reward $R_{t+1} \in \mathbb{R}$. The agent's objective is to evaluate a fixed *target policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, or estimate the expected return for policy π:

$$v^\pi(s) \stackrel{\text{def}}{=} \mathbb{E}[G_t | S_t = s, A_t \sim \pi]$$

$$\text{for return } G_t \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} \left(\prod_{j=1}^{i} \gamma_{t+j} \right) R_{t+i+1} \qquad \triangleright \gamma_j \stackrel{\text{def}}{=} \gamma(s_j).$$

where $v^\pi(s)$ is called the *state-value function* for policy π.

All the methods evaluated in this study perform temporal difference updates, and most utilize eligibility traces. The TD(λ) algorithm is the prototypical example of these concepts and is useful for understanding all the other algorithms discussed in the remainder of this paper. TD(λ) estimates v^π as a linear function of the weight vector $\mathbf{w} \in \mathbb{R}^d$, where the estimate is formed as an inner product between the weight vector and the features of the current state: $\mathbf{w}^\top \mathbf{x}(s) \approx v^\pi(s)$. The algorithm maintains a memory trace of recently experienced features, called the eligibility trace $\mathbf{e} \in \mathbb{R}^d$, allowing updates to assign credit to previously visited states. The TD(λ) algorithm requires linear computation and storage $O(d)$, and can be implemented incrementally as follows:

$$\delta_t \leftarrow R_{t+1} + \gamma_{t+1} \mathbf{w}_t^\top \mathbf{x}(S_{t+1}) - \mathbf{w}_t^\top \mathbf{x}(S_t)$$
$$\mathbf{e}_t \leftarrow \lambda_t \gamma_t \mathbf{e}_{t-1} + \mathbf{x}(S_t)$$
$$\Delta \mathbf{w} \leftarrow \alpha \delta_t \mathbf{e}_t \qquad \triangleright \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Delta \mathbf{w}.$$

In the case when the data is generated by a behavior policy, μ, with $\pi \neq \mu$, we say that the data is generated off-policy. In the off-policy setting we must estimate v^π with samples generated by selecting actions according to μ. This setting can cause the TD(λ) algorithm to diverge. The GTD(λ) algorithm solves the divergence issue by minimizing the MSPBE, resulting in a stochastic gradient descent algorithm that looks similar to TD(λ), with some important differences. GTD(λ) uses importance weights, $\rho_t \stackrel{\text{def}}{=} \frac{\pi(s,a)}{\mu(s,a)} \in \mathbb{R}$ in the eligibility trace to reweight the data and obtain an unbiased estimate of $\mathbb{E}[G_t]$. Note, in the policy iteration case—not studied here—it is still reasonable to assume knowledge of $\pi(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$; for example when π is near greedy with respect to the current estimate of the state-action value function. The GTD(λ) has a auxiliary set of learned weights, $\mathbf{h} \in \mathbb{R}^d$, in addition to the primary weights $\mathbf{w}$, which maintain a quasi-stationary estimate of a part of the MSPBE. Like the TD(λ) algorithm, GTD(λ)

requires only linear computation and storage and can be implemented fully incrementally as follows:

$$\delta_t \leftarrow R_{t+1} + \gamma_{t+1}\mathbf{w}_t^\top \mathbf{x}(S_{t+1}) - \mathbf{w}_t^\top \mathbf{x}(S_t)$$
$$\mathbf{e}_t \leftarrow \rho_t(\lambda_t\gamma_t\mathbf{e}_{t-1} + \mathbf{x}(S_t)) \qquad \triangleright \text{ weighted by } \rho_t$$
$$\Delta\mathbf{w} \leftarrow \alpha\delta_t\mathbf{e}_t - \underbrace{\alpha\gamma_{t+1}(1-\lambda_{t+1})(\mathbf{e}_t^\top\mathbf{h}_t)\mathbf{x}(S_{t+1})}_{\text{correction term}}$$
$$\Delta\mathbf{h} \leftarrow \alpha_\mathbf{h}[\delta_t\mathbf{e}_t - (\mathbf{x}(S_t)^\top\mathbf{h}_t)\mathbf{x}(S_t)] \qquad \triangleright \text{auxiliary weights}$$

The auxiliary weights also make use of a step-size parameter, $\alpha_\mathbf{h}$ which is usually not equal to α.

Due to space constraints we do not describe the other TD-based linear learning algorithms found in the literature and investigated in our study. We provide each algorithm's pseudo code in the appendix, and in the next section describe two new off-policy, gradient TD methods, before turning to empirical questions.

3. HTD DERIVATION

Conventional temporal difference updating can be more data efficient than gradient temporal difference updating, but the correction term used by gradient-TD methods helps prevent divergence. Previous empirical studies[17] demonstrated situations (specifically on-policy) where linear TD(0) can outperform gradient TD methods, and others [6] demonstrated that Expected Sarsa(0) can outperform multiple variants of the GQ(0) algorithm, even under off-policy sampling. On the other hand, TD(λ) can diverge on small, though somewhat contrived counterexamples.

The idea of hybrid-TD methods is to achieve sample efficiency closer to TD(λ) during on-policy sampling, while ensuring non-divergence under off-policy sampling. To achieve this, a hybrid algorithm could do conventional, uncorrected TD updates when the data is sampled on-policy, and use gradient corrections when the data is sampled off-policy. This approach was pioneered by Maei [11], leading to the derivation of the Hybrid Temporal Difference learning algorithm, or HTD(0). Later, Hackman[6] produced a hybrid version of the GQ(0) algorithm, estimating state-action value functions rather than state-value functions as we do here. In this paper, we derive the first hybrid temporal difference method to make use of eligibility traces, called HTD(λ).

The key idea behind the derivation of HTD learning methods is to modify the gradient of the MSPBE to produce a new learning algorithm. Let $\mathbb{E}_\mu$ represent the expectation according to samples generated under the behavior policy, μ. The MSPBE[17] can be written as

$$\texttt{MSPBE}(\mathbf{w}) = \underbrace{\mathbb{E}_\mu[\delta_t\mathbf{e}_t]^\top}_{-A_\pi\mathbf{w}+b_\pi}\underbrace{\mathbb{E}_\mu[\mathbf{x}(S_t)\mathbf{x}(S_t)^\top]^{-1}}_{C}\mathbb{E}_\mu[\delta_t\mathbf{e}_t],$$

where $\mathbf{e}_t = \rho_t(\lambda_t\gamma_t\mathbf{e}_{t-1} + \mathbf{x}(S_t))$ and

$$A_\pi \overset{\text{def}}{=} \mathbb{E}_\mu[\mathbf{e}_t(\mathbf{x}(S_t) - \gamma_{t+1}\mathbf{x}(S_{t+1}))^\top] \tag{1}$$
$$= \sum_{s_t\in\mathcal{S}} d^\mu(s_t) \sum_{a_t\in\mathcal{A}} \underbrace{\mu(s_t,a_t)}_{\pi(s_t,a_t)}\rho_t(\gamma_t\lambda\mathbb{E}_\mu[\mathbf{e}_{t-1}|s_t] + \mathbf{x}(s_t))$$
$$\sum_{s_{t+1}\in\mathcal{S}} P(s_t,a_t,s_{t+1})(\mathbf{x}(s_t) - \gamma_{t+1}\mathbf{x}(s_{t+1}))^\top$$

$$b_\pi \overset{\text{def}}{=} \mathbb{E}_\mu[R_{t+1}\mathbf{e}_t]$$
$$= \sum_{s_t\in\mathcal{S}} d^\mu(s_t) \sum_{a_t\in\mathcal{A}} \pi(s_t,a_t)(\gamma_t\lambda\mathbb{E}_\mu[\mathbf{e}_{t-1}|s_t] + \mathbf{x}(s_t))$$
$$\sum_{s_{t+1}\in\mathcal{S}} \pi(s_t,a_t)P(s_t,a_t,s_{t+1})r_{t+1}.$$

Therefore, the relative importance given to states in the MSPBE is weighted by the stationary distribution of the behavior policy, $d_\mu : \mathcal{S} \to \mathbb{R}$, (since it is generating samples), but the transitions are reweighted to reflect the returns that π would produce.

The gradient of the MSPBE is:

$$-\frac{1}{2}\nabla_\mathbf{w}\texttt{MSPBE}(\mathbf{w}) = -A_\pi^\top C^{-1}(-A_\pi\mathbf{w} + b_\pi). \tag{2}$$

Assuming A_π^{-1} is non-singular, we get the TD-fixed point solution:

$$0 = -\frac{1}{2}\nabla_\mathbf{w}\texttt{MSPBE}(\mathbf{w}) \implies -A_\pi\mathbf{w} + b_\pi = 0. \tag{3}$$

The value of $\mathbf{w}$, for which (3) is zero, is the solution found by linear TD(λ) and LSTD(λ) where $\pi = \mu$. The gradient of the MSPBE yields an incremental learning rule with the following general form (see [3]):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(M\mathbf{w}_t + b), \tag{4}$$

where $M = -A_\pi^\top C^{-1}A_\pi$ and $b = A_\pi^\top C^{-1}b_\pi$. The update rule, in the case of TD(λ), will yield stable convergence if A_π is positive definite (as shown by Tsitsiklis and van Roy [22]). In off-policy learning, we require $A_\pi^\top C^{-1}A_\pi$ to be positive definite to satisfy the conditions of the ordinary differential equation proof of convergence [10], which holds because C^{-1} is positive definite and therefore $A_\pi^\top C^{-1}A_\pi$ is positive definite, because A_π is full rank (true by assumption). See Sutton et al. [21] for a nice discussion on why the A_π matrix must be positive definite to ensure stable, non-divergent iterations. The C matrix in Equation (3), can be replaced by any positive definite matrix and the fixed point will be unaffected, but the rate of convergence will almost surely change.

Instead of following the usual recipe for deriving GTD, let us try replacing C^{-1} with

$$A_\mu^{-\top} \overset{\text{def}}{=} \mathbb{E}_\mu[(\mathbf{x}(S_t) - \gamma_t\mathbf{x}(S_{t+1}))\mathbf{e}_t^{\mu\top}],$$

where $\mathbf{e}^\mu$ is the regular on-policy trace for the behavior policy (i.e., no importance weights)

$$\mathbf{e}_t^\mu = \gamma_t\lambda\mathbf{e}_{t-1}^\mu + \mathbf{x}(S_t).$$

The matrix $A_\mu^{-\top}$ is a positive definite matrix (proved by Tsitsiklis and van Roy [22]). Plugging $A_\mu^{-\top}$ into (2) results in the following expected update:

$$\frac{1}{\alpha}\mathbb{E}[\Delta\mathbf{w}_t] = A_\pi^\top A_\mu^{-\top}(-A_\pi\mathbf{w}_t + b_\pi)$$
$$= (A_\mu^\top - A_\mu^\top + A_\pi^\top)A_\mu^{-\top}(-A_\pi\mathbf{w}_t + b_\pi)$$
$$= (A_\mu^\top A_\mu^{-\top})(-A_\pi\mathbf{w}_t + b_\pi) + (A_\pi^\top - A_\mu^\top)A_\mu^{-\top}(-A_\pi\mathbf{w}_t + b_\pi)$$
$$= (-A_\pi\mathbf{w}_t + b_\pi) + (A_\pi^\top - A_\mu^\top)A_\mu^{-\top}(-A_\pi\mathbf{w}_t + b_\pi)$$
$$= (-A_\pi\mathbf{w}_t + b_\pi) + \tag{5}$$
$$\mathbb{E}_\mu\left[(\mathbf{x}(S_t) - \gamma_{t+1}\mathbf{x}(S_{t+1}))(\mathbf{e}_t - \mathbf{e}_t^\mu)^\top\right]A_\mu^{-\top}(-A_\pi\mathbf{w}_t + b_\pi)$$

As in the derivation of GTD(λ) [11], let the vector $\mathbf{h}_t$ form a quasi-stationary estimate of the final term,

$$A_\mu^{-\top}(-A_\pi \mathbf{w}_t + b_\pi).$$

Getting back to the primary weight update, we can sample the first term using the fact that $(-A_\pi \mathbf{w}_t + b_\pi) = \mathbb{E}_\mu[\delta_t \mathbf{e}_t]$ (see [11]) and use (1) to get the final stochastic update

$$\Delta \mathbf{w}_t \leftarrow \alpha \left(\delta_t \mathbf{e}_t + (\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1}) \left(\mathbf{e}_t - \mathbf{e}_t^\mu\right)^\top \mathbf{h}_t \right). \quad (6)$$

Notice that when the data is generated on-policy ($\pi = \mu$), $\mathbf{e}_t = \mathbf{e}_t^\mu$, and thus the correction term disappears and we are left with precisely linear TD(λ). When $\pi \neq \mu$, the TD update is corrected as in GTD: unsurprisingly, the correction is slightly different but has the same basic form.

To complete the derivation, we must derive an incremental update rule for $\mathbf{h}_t$. We have a linear system, because

$$\mathbf{h}_t = A_\mu^{-\top}(-A_\pi \mathbf{w}_t + b_\pi) \implies A_\mu^\top \mathbf{h}_t = -A_\pi \mathbf{w}_t + b_\pi.$$

Following the general expected update in (4),

$$\mathbf{h}_{t+1} \leftarrow \mathbf{h}_t + \alpha_\mathbf{h} \left((-A_\pi \mathbf{w}_t + b_\pi) - A_\mu^\top \mathbf{h}_t \right) \quad (7)$$

which converges if $A_\mu^\top$ is positive definite for any fixed $\mathbf{w}_t$ and $\alpha_\mathbf{h}$ is chosen appropriately (see Sutton et al.'s recent paper[21] for an extensive discussion of convergence in expectation). To sample this update, recall

$$A_\mu^\top \mathbf{h}_t = \mathbb{E}_\mu[(\mathbf{x}(S_t) - \mathbf{x}(S_{t+1}))\mathbf{e}_t^{\mu\top}]\mathbf{h}_t$$

giving stochastic update rule for $\mathbf{h}_t$:

$$\Delta \mathbf{h}_t \leftarrow \alpha_\mathbf{h} \left[\delta_t \mathbf{e}_t - (\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1}) \mathbf{e}_t^{\mu\top} \mathbf{h}_t \right].$$

As in GTD, $\alpha \in \mathbb{R}$ and $\alpha_\mathbf{h} \in \mathbb{R}$ are step-size parameters, and $\delta_t \overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}\mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t$. This hybrid-TD algorithm should converge under off-policy sampling using a proof technique similar to the one used for GQ(λ) (see Maei & Sutton's proof [10]), but we leave this to future work. The HTD(λ) algorithm is completely specified by the following equations:

$$\begin{aligned}
\mathbf{e}_t &\leftarrow \rho_t(\lambda_t \gamma_t \mathbf{e}_{t-1} + \mathbf{x}_t) \\
\mathbf{e}_t^\mu &\leftarrow \lambda_t \gamma_t \mathbf{e}_{t-1}^\mu + \mathbf{x}_t \\
\Delta \mathbf{w}_t &\leftarrow \alpha \left[\delta_t \mathbf{e}_t + (\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)(\mathbf{e}_t^\mu - \mathbf{e}_t)^\top \mathbf{h}_t \right] \\
\Delta \mathbf{h}_t &\leftarrow \alpha_\mathbf{h} \left[\delta_t \mathbf{e}_t + (\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{e}_t^{\mu\top} \mathbf{h}_t \right]
\end{aligned}$$

This algorithm can be made more efficient by exploiting the common terms in $\Delta \mathbf{w}_t$ and $\Delta \mathbf{h}_t$, as shown in the appendix.

4. TRUE ONLINE HTD

Recently, a new forward-backward view equivalence has been proposed for online TD methods, resulting in true-online TD [24] and true-online GTD [23] algorithms. The original forward-backward equivalence was for offline TD(λ)[2]. To derive a forward-backward equivalence under online updating, a new truncated return was proposed, which uses the online weight vector that changes into the future,

$$G_{k,t}^{\lambda,\rho} \overset{\text{def}}{=} \rho_k(R_{k+1} + \gamma_{k+1}[(1-\lambda_{k+1})\mathbf{x}_{t+1}^\top \mathbf{w}_k + \lambda_{k+1} G_{k+1,t}^{\lambda,\rho}]),$$

with $G_{t,t}^{\lambda,\rho} \overset{\text{def}}{=} \rho_t \mathbf{x}_t^\top \mathbf{w}_{t-1}$. A forward-view algorithm can be defined that computes $\mathbf{w}_k$ online assuming access to future samples, and then an exactly equivalent incremental backward-view algorithm can be derived that does not require access to future samples. This framework was used to derive the TO-TD algorithm for the on-policy setting, and TO-GTD for the more general off-policy setting. This new true-online equivalence is not only interesting theoretically, but also translates into improved prediction and control performance [24, 23].

In this section, we derive a true-online variant of HTD(λ). When used on-policy HTD(λ) behaves similarly to TO-TD(λ). Our goal in this section is to combine the benefits of both hybrid learning and true-online traces in a single algorithm. We proceed with a similar derivation to TO-GTD(λ) [23, Theorem 4], with the main difference appearing in the update of the auxiliary weights. Notice that the primary weights $\mathbf{w}$, and the auxiliary weights $\mathbf{h}$, of HTD(λ) have a similar structure. Recall from (5), the modified gradient of the MSPBE, or expected primary-weight update can be written as:

$$\begin{aligned}
\frac{1}{\alpha}\mathbb{E}[\Delta \mathbf{w}_t] &= (-A_\pi \mathbf{w}_t + b_\pi) \\
&\quad + \mathbb{E}_\mu \left[(\mathbf{x}(S_t) - \gamma_{t+1}\mathbf{x}(S_{t+1})) \left(\mathbf{e}_t - \mathbf{e}_t^\mu\right)^\top \right] \mathbf{h}_t
\end{aligned}$$

Similarly, we can rewrite the expected update of the auxiliary weights by plugging $A_\mu^\top$ into (7):

$$\begin{aligned}
\frac{1}{\alpha_h}\mathbb{E}[\Delta \mathbf{h}_t] &= (-A_\pi \mathbf{w}_t + b_\pi) \\
&\quad + \mathbb{E}_\mu \left[(\mathbf{x}(S_t) - \gamma_{t+1}\mathbf{x}(S_{t+1})) \, \mathbf{e}_t^{\mu\top} \right] \mathbf{h}_t
\end{aligned}$$

As in the derivation of TO-GTD [23, Equation 17,18]), for TO-HTD we will sample the second part of the update using a backward-view and obtain forward-view samples for $(-A_\pi \mathbf{w}_t + b_\pi)$. The resulting TO-HTD($\lambda$) algorithm is completely specified by the following equations

$$\begin{aligned}
\mathbf{e}_t &\leftarrow \rho_t(\lambda_t \gamma_t \mathbf{e}_{t-1} + \mathbf{x}_t) \\
\mathbf{e}_t^\mu &\leftarrow \lambda_t \gamma_t \mathbf{e}_{t-1}^\mu + \mathbf{x}_t \\
\mathbf{e}_t^o &\leftarrow \rho_t(\lambda_t \gamma_t \mathbf{e}_{t-1}^o + \alpha_t(1 - \rho_t \gamma_t \lambda_t \mathbf{x}_t^\top \mathbf{e}_{t-1}^o)\mathbf{x}_t) \\
\mathbf{d} &= \delta_t \mathbf{e}_t^o + (\mathbf{e}_t^o - \alpha_t \rho_t \mathbf{x}_t)(\mathbf{w}_t - \mathbf{w}_{t-1})^\top \mathbf{x}_t \quad (8) \\
\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \mathbf{d} + \alpha_t(\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)(\mathbf{e}_t^\mu - \mathbf{e}_t)^\top \mathbf{h}_t \\
\mathbf{h}_{t+1} &\leftarrow \mathbf{h}_t + \mathbf{d} + \alpha_\mathbf{h}(\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{e}^{\mu\top}{}_t \mathbf{h}_t
\end{aligned}$$

In order to prove that this is a true-online update, we use the constructive theorem due to van Hasselt et al. [23].

THEOREM 1 (TRUE-ONLINE HTD(λ)). *For any t, the weight vectors $\mathbf{w}_t^t, \mathbf{h}_t^t$ as defined by the forward view*

$$\begin{aligned}
\mathbf{w}_{k+1}^t &= \mathbf{w}_k^t + \alpha_k(G_{k,t}^{\lambda,\rho} - \rho_k \mathbf{x}_t^\top \mathbf{w}_k^t)\mathbf{x}_k \\
&\quad + \alpha_k(\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1})(\mathbf{e}_t - \mathbf{e}_t^\mu)^\top \mathbf{h}_k^t
\end{aligned}$$

$$\begin{aligned}
\mathbf{h}_{k+1}^t &= \mathbf{h}_k^t + \alpha_{h,k}(G_{k,t}^{\lambda,\rho} - \rho_k \mathbf{x}_t^\top \mathbf{w}_k^t)\mathbf{x}_k \\
&\quad + \alpha_{h,k}(\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1})\mathbf{e}_t^{\mu\top} \mathbf{h}_k^t
\end{aligned}$$

are equal to $\mathbf{w}_t$, $\mathbf{h}_t$ as defined by the backward view in (8).

[2]The idea of defining a forward view objective and then converting this computationally impractical forward-view into an efficiently implementable algorithm using traces is extensively treated in Sutton and Barto's introductory text [16].

PROOF. We apply [23, Theorem 1]. The substitutions are

$$\eta_t = \rho_t \alpha_t$$
$$\mathbf{g}_{w,k} = \alpha_k(\mathbf{x}_k - \gamma_{k+1}\mathbf{x}_{k+1})(\mathbf{e}_k - \mathbf{e}_k^\mu)^\top \mathbf{h}_k$$
$$\mathbf{g}_{h,k} = \alpha_{h,k}(\mathbf{x}_k - \gamma_{k+1}\mathbf{x}_{k+1})\mathbf{e}_k^{\mu\top} \mathbf{h}_k$$
$$Y_t^t = \mathbf{w}_{t-1}^\top \mathbf{x}_t$$
$$Y_k^t = R_{k+1} + \gamma_{k+1}(1 - \lambda_{k+1}\rho_{k+1})\mathbf{w}_k^\top \mathbf{x}_{k+1} + \gamma_{k+1}\lambda_{k+1} G_{k+1,t}^{\lambda,\rho}$$

where $\mathbf{g}_{w,k}$ is called $\mathbf{x}_k$ in van Hasselt's Theorem 1 [23]. The proof then follows through in the same way as in van Hasselt's Theorem 4 [23], where we apply Theorem 1 to $\mathbf{w}$ and $\mathbf{h}$ separately. □

Our TO-HTD(0) algorithm is equivalent to HTD(0), but TO-HTD(λ) is not equivalent to TO-TD(λ) under on-policy sampling. To achieve the later equivalence, replace $\delta_t \stackrel{\text{def}}{=} R_{t+1} + \gamma_{t+1}\mathbf{w}_t^\top \mathbf{x}_{t+1} + \mathbf{w}_{t-1}^\top \mathbf{x}_t$ and $\mathbf{d} \stackrel{\text{def}}{=} \delta_t \mathbf{e}_t^o - \alpha_t \rho_t \mathbf{x}_t(\mathbf{w}_t - \mathbf{w}_{t-1})^\top \mathbf{x}_t$. We opted for the first equivalence for two reasons. In preliminary experiments, TO-HTD(λ) described in Equation (8) already exhibited similar performance compared to TO-TD(λ), and so designing for the second equivalence was unnecessary. Further, TO-GTD(λ) was derived to ensure equivalence between TO-GTD(0) and GTD(0); this choice, therefore, better parallels that equivalence.

Given our two new hybrid methods, and the long list of existing linear prediction algorithms we now focus on how these algorithms perform in practice.

5. EXPERIMENTAL STUDY

Our empirical study focused on three main aspects: (1) early learning performance with different feature representations, (2) parameter sensitivity, and, (3) efficacy in on and off-policy learning. The majority of our experiments were conducted on random MDPs (variants of those used in previous studies[14, 5]). Each random MDP contains 30 states, and three actions in each state. From each state, and for each action, the agent can transition to one of four next states, assigned randomly from the entire set without replacement. Transition probabilities for each MDP instance are randomly sampled from $[0, 1]$ and the transitions were normalized to sum to one. The expected reward for each transition is also generated randomly in $[0, 1]$ and the reward on each transition was sampled without noise. Two transitions are randomly selected to terminate: $\gamma(s_i, s_j) = 0$ for $i \neq j$. Each problem instance is held fixed during learning.

We experimented with three different feature representations. The first, a *tabular* representation where each state is represented with a binary vector with a single one corresponding the current state index. This encoding allows perfect representation of the value function with no generalization over states. The second representation is computed by taking the tabular representation and *aliasing* five states to all have the same feature vector, so the agent cannot differentiate these states. These five states were selected randomly without replacement for each MDP instance. The third representation is a dense *binary* encoding where the feature vector for each state is the binary encoding of the state index, and thus the feature vector for a 30 state MDP requires just five components. Although the binary representation appears to exhibit an inappropriate amount of generalization, we believe it to be more realistic that a tabular representation, because access to MDP state is rare in real-world domains (e.g., a robotic with continuous sensor values). The binary representation should be viewed as an approximation to the poor, and relatively low-dimensional (compared to the number of states in the world) representations common in real applications. All feature encoding we normalized. Experiments conducted with the binary representation use $\gamma = 0.99$, and the rest use $\gamma = 0.9$.

To generate policies with purposeful behavior, we forced the agent to favor a single action in each state. The target policy is generated by randomly selecting an action and assigning it probability 0.9 (i.e., $\pi(s, a_i) = 0.9$) in each state, and then assigning the remaining actions the remaining probability evenly. In the off-policy experiments the behavior policy is modified to be slightly different than the target policy, by selecting the same base action, but instead assigning a probability of 0.8 (i.e., $\mu(s, a_i) = 0.8$) . This choice ensures that the policies are related, but guarantees that ρ_t is never greater than 1.5 thus avoiding inappropriately large variance due to importance sampling[3].

Our experiment compared 12 different linear complexity value function learning algorithms, including: GTD(λ), HTD(λ), true-online GTD(λ), true-online HTD(λ), true-online ETD(λ), true-online ETD(λ, β), PTD(λ), GTD2 - mp(λ), TDC - mp(λ), linear off-policy TD(0), TD(λ), true-online TD(λ). The later two being only applicable in on-policy domains, and the two mirror-prox methods are straightforward extensions (and described in the appendix) of the GTD2-mp and TDC-mp methods [13] to handle traces ($\lambda > 0$). We drop the λ designation of each method in the figure labels to reduce clutter.

Our results were generated by performing a large parameter sweep, averaged over many independent runs, for each random MDP instance, and then averaging the results over the entire set of MDPs. We tested 14 different values of the step-size parameter $\alpha \in \{0.1 \times 2^j | j = -8, -7, ..., 6\}$, seven values of $\eta \in \{2^j | j = -4, -2, -1, 0, 1, 2, 4\}$ ($\alpha_\mathbf{h} \stackrel{\text{def}}{=} \alpha\eta$), and 20 values of $\lambda = \{0, 0.1, ..., 0.9, 0.91, ..., 1.0\}$. We intentionally precluded smaller values of $\alpha_\mathbf{h}$ from the parameter sweep because many of the gradient TD methods simply become their on-policy variants as $\alpha_\mathbf{h}$ approaches zero, whereas in some off-policy domains values of $\alpha_\mathbf{h} > \alpha$ are required to avoid divergence [26]. We believe this range of η fairly reflects how the algorithms would be used in practice if avoiding divergence was a priority. The β parameter of TO-ETD(λ, β) was set equal to $0.5\gamma_t$. Each algorithm instance, defined by one combination of α, η, and λ was evaluated using the mean absolute value error on each time step,

$$\epsilon_t \stackrel{\text{def}}{=} \sum_{s \in \mathcal{S}} d_\mu(s) \left| \frac{\mathbf{x}(s)^\top \mathbf{w}_t - V^*(s)}{V^*(s)} \right|,$$

averaged over 30 MDPs, each with 100 runs. Here $V^* : \mathcal{S} \to \mathbb{R}$ denotes the *true* state-value function, which can be easily computed with access to the parameters of the MDP.

The graphs in Figures 1 and 2 include (a) learning curves with α, η, and λ selected to minimize the mean absolute value error, for each of the three different feature representations, and (b) parameter sensitivity graphs for α, η, and λ, in which the mean absolute value error is plotted against the parameter value, while the remaining two parameters

[3]See the recent study by Mahmood & Sutton [14] for an extensive treatment of off-policy learning domains with large variance due to importance sampling.

are selected to minimize mean absolute value error. These graphs are included across feature representations, for on and off-policy learning. Across all results the parameters are selected to optimize performance over the last half of the experiment to ensure stable learning throughout the run.

To analyze large variance due to importance sampling and off-policy learning we also investigated Baird's counterexample [2], a simple MDP that causes TD learning to diverge. This seven state MDP uses a target policy that is very different from the behavior policy, a feature representation that allows perfect representation of the value function, but also causes inappropriate generalization. We used the variant of this problem described by Maei [11] and White [26, Figure 7.1]. We present results with the root mean squared error [4],

$$\epsilon_t \stackrel{\text{def}}{=} \sum_{s \in \mathcal{S}} d_\mu(s) \left(\mathbf{x}(s)^\top \mathbf{w}_t - V^*(s) \right)^2,$$

in Figure 1. The experiment was conducted in the same way was the random MDPs, except we did not average over MDPs—there is only one—and we used different parameter ranges. We tested 11 different values of the step-size parameter $\alpha \in \{0.1 \times 2^j | j = -10, -9, ..., -1, 0\}$, 12 values of $\eta \in \{2^j | j = -16, -8, \ldots, -2, -1, 0, 1, 2, \ldots, 32\}$ ($\alpha_\mathbf{h} \stackrel{\text{def}}{=} \alpha\eta$), and the same 20 values of λ. We did not evaluate TD(0) on this domain because the algorithm will diverge and that has been shown many times before.

In addition to performance results in Figures 1 and 2, Table 1 summarizes the runtime comparison for these algorithms. Though the algorithms are all linear in storage and computation, they do differ in both implementation and runtime, particularly due to true-online traces. The appendix contains several plots of runtime verses value error illustrating the trade-off between computation and sample complexity for each algorithm. Due to space constraints, we have included the aliased tabular representation results for on-policy learning in the appendix, since they are similar to the tabular representation results in on-policy learning.

6. DISCUSSION

There are three broad conclusions suggested by our results. First, we could not clearly demonstrate the supposed superiority of TD(λ) over gradient TD methods in the on-policy setting. In both tabular and aliased feature settings GTD(λ) achieved faster learning and superior parameter sensitivity compared to TD, PTD, and HTD. Notably, the η-sensitivity of the GTD algorithm was very reasonable in both domains, however, large η were required to achieve good performance on Baird's for both GTD(λ) and TO-GTD(λ). Our on-policy experiments with binary features did indicate a slight advantage for TD(λ), PTD, and HTD, and that PTD and HTD exhibit zero sensitivity to the choice of $\alpha_\mathbf{h}$ as expected. In off-policy learning there is little difference between GTD(λ) and PTD and HTD. Our results combined with the prior work of Dann et al. [4] suggest that the advantage of conventional TD(λ) over gradient TD methods, in on-policy learning, is limited to specific domains.

[4]In this counterexample the mean absolute value error is not appropriate because the optimal values for this task are zero. The MSPBE is often used as a performance measure, but the MSPBE changes with λ; for completeness, we include results with the MSPBE in the appendix.

Our second conclusion, is that the new mirror prox methods achieved poor performance in most settings except Baird's counterexample. Both GTD2-mp and TDC-mp achieved the best performance in Baird's counterexample. We hypothesize that the two-step gradient computation more effectively uses the transition to state 7, and so is ideally suited to the structure of the domain[5]. However, the GTD2-MP method performed worse than off-policy TD(0) in all off-policy random MDP domains, while the learning curves of TDC-mp exhibited higher variance than other methods in all but the on-policy binary case and high parameter sensitivity across all settings except Baird's. This does not seem to be a consequence of the extension to eligibility traces because in all cases except Baird's, both TDC-mp and GTD2-mp performed best with $\lambda > 0$. Like GTD and HTD, the mirror prox methods would likely have performed better with values of $\alpha_\mathbf{h} > \alpha$, however, this is undesirable because larger $\alpha_\mathbf{h}$ is required to ensure good performance in some off policy domains, such as Baird's (e.g., $\eta = 2^8$).

Third and finally, several methods exhibited non-convergent behavior on Baird's counterexample. All methods that exhibited reliable error reduce in Baird's did so with λ near zero, suggesting that eligibility traces are of limited use in these more extreme off-policy domains. In the case of PTD, non-convergent behavior is not surprising since our implementation of this algorithm does not include gradient correction—a possible extension suggested by the authors [20]—and thus is only guaranteed to converge under off-policy sampling in the tabular case. For the emphatic TD methods the performance on Baird's remains a concern, especially considering how well TO-ETD(λ, β) performed in all our other experiments. The addition of the β parameter appears to significantly improve TO-ETD in the off-policy domain with binary features, but could not mitigate the large variance in ρ produced by the counterexample. It is not clear if this bad behavior is inherent to emphatic TD methods[6], or could be solved by more careful specification of the state-based interest function. In our implementation, we followed the original author's recommendation of setting the interest for each state to 1.0 [21], because all our domains were discounted and continuing. Additionally, both HTD(λ) and TO-HTD(λ) did not diverge on Baird's, but performance was less than satisfactory to say the least.

Overall, the conclusions implied by our empirical study are surprisingly clear. If guarding against large variance due to off-policy sampling is a chief concern, then GTD(λ) and TO-GTD(λ) should be preferred. Between the two, GTD(λ) should be preferred if computation is at a premium. If poor performance in problems like Baird's is not a concern, then TO-ETD(λ, β) was clearly the best across our experiments, and exhibited nearly the best runtime results. TO-ETD(λ) on the other hand, exhibited high variance in off-policy domains, and sharp parameter sensitivity, indicating parameter turnng of emphatic methods may be an issue in practice.

7. APPENDIX

Additional results and analysis can be found in the full version of the paper: http://arxiv.org/abs/1602.08771.

[5]Baird's counterexample uses a specific initialization of the primary weights: far from one of the true solutions $\mathbf{w} = \mathbf{0}$.
[6]The variance of TO-ETD has been examined before in two state domains [21]. ETD is thought to have higher variance that other TD algorithms due to the emphasis weighting.

	TD(0)	TD(λ)	TO-TD	PTD	GTD	TO-ETD	TO-ETD(β)	HTD	TO-GTD	GTD-MP	TDC-MP	TO-HTD
On-policy	120.0	132.7	150.1	172.4	204.6	287.8	286.0	311.8	366.2	467.4	466.2	466.0
Off-policy	108.3	-	-	158.7	175.2	249.65	254.7	267.5	316.2	407.8	395.7	403.3

Table 1: Average runtime in microseconds for 500 steps of learning, averaged over 30 MDPS, with 100 runs each, with 30-dimensional tabular features.

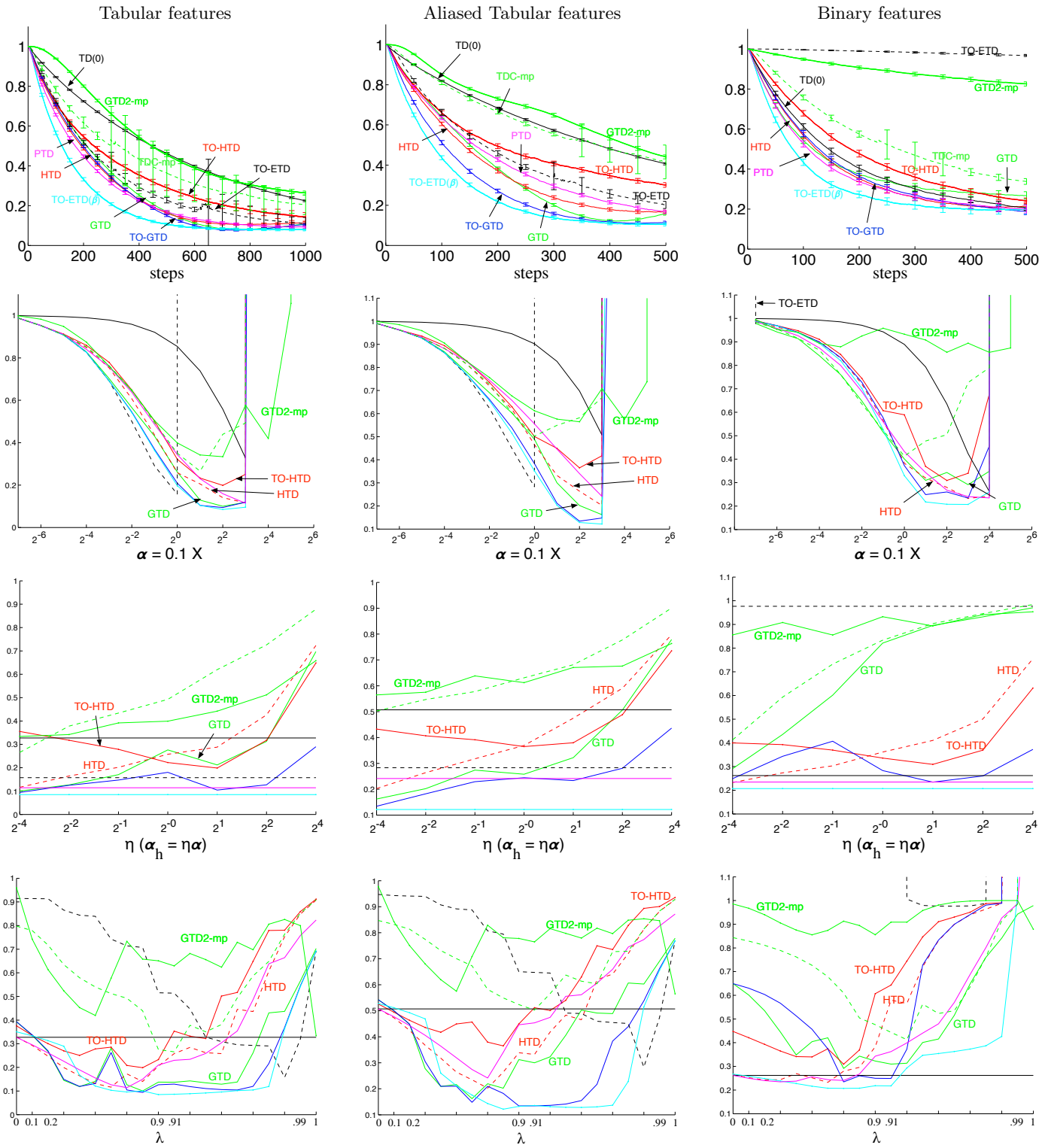

Figure 1: **Off-policy** performance on random MDPs with three different representations. All plots report mean absolute value error averaged over 100 runs and 30 MDPs. The plots are organized in columns left to right corresponding to tabular, aliased, and binary features. The plots are organized in rows from top to bottom corresponding to learning curves, α, η, and λ sensitivity. The error bars are standard errors ($s/\sqrt{(n)}$) computed from 100 independent runs.

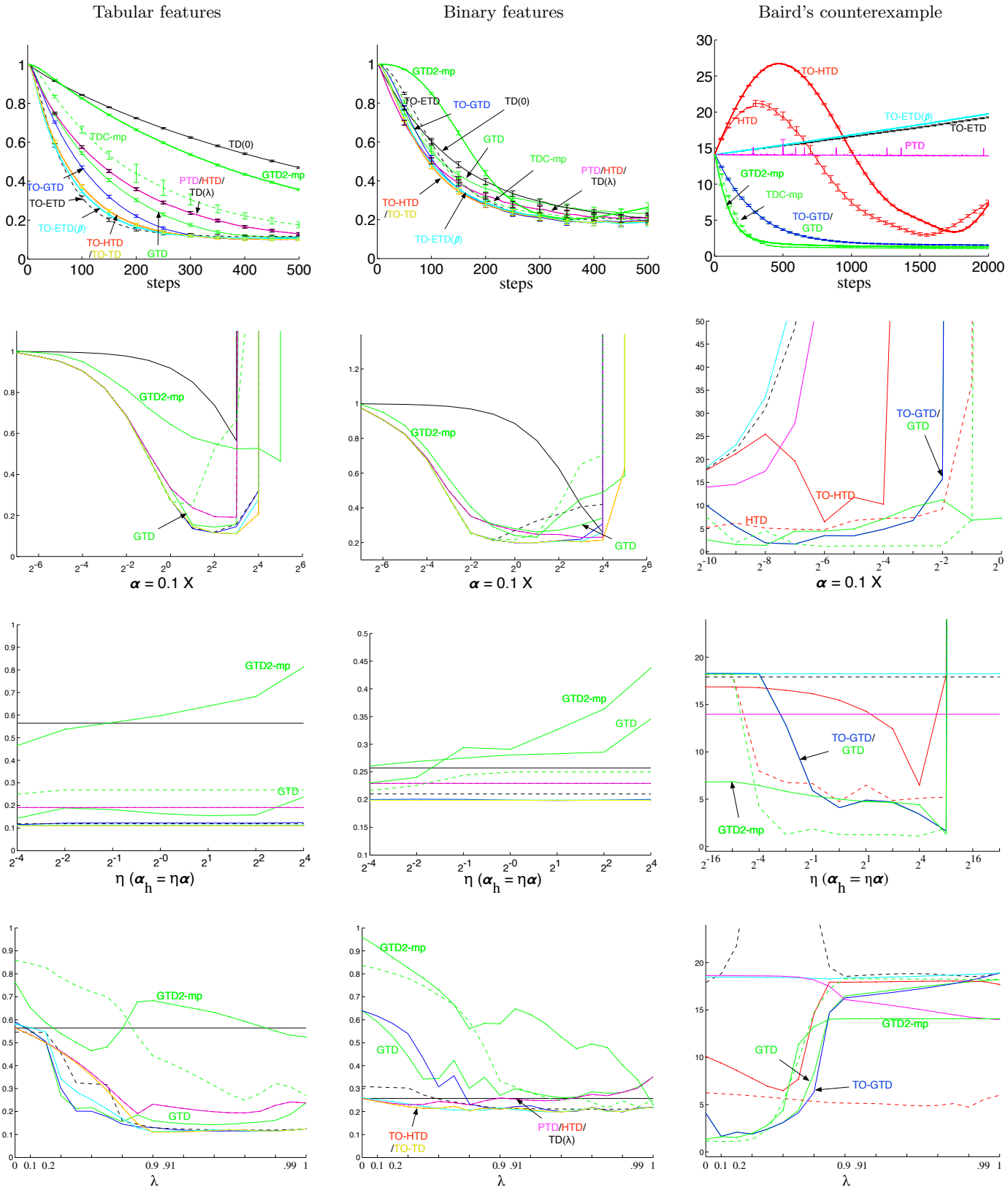

Figure 2: **On-policy** performance on random MDPs with two different representations and **off-policy** performance on Baird's counterexample. All plots report mean absolute value error averaged over 100 runs and 30 random MDPs, and 500 runs for Baird's. The plots are organized in columns left to right corresponding to results on random MDPs with tabular and binary features, and results on Baird's counterexample. The plots are also organized in rows from top to bottom corresponding to learning curves, α, η, and λ sensitivity.

REFERENCES

[1] B. Argall, S. Chernova, M. M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems ()*, 2009.

[2] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, 1995.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific Press, 1996.

[4] C. Dann, G. Neumann, and J. Peters. Policy evaluation with temporal differences: a survey and comparison. *The Journal of Machine Learning Research*, 2014.

[5] M. Geist and B. Scherrer. Off-policy learning with eligibility traces: a survey. *The Journal of Machine Learning Research*, 2014.

[6] L. Hackman. *Faster Gradient-TD Algorithms*. PhD thesis, University of Alberta, 2012.

[7] A. Hallak, A. Tamar, R. Munos, and S. Mannor. Generalized emphatic temporal difference learning: Bias-variance analysis. *arXiv preprint arXiv:1509.05172*, 2015.

[8] L.-J. Lin. Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Machine Learning*, 1992.

[9] B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik. Finite-Sample Analysis of Proximal Gradient TD Algorithms. *Conference on Uncertainty in Artificial Intelligence*, 2015.

[10] H. Maei and R. Sutton. GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *AGI*, 2010.

[11] H. R. Maei. *Gradient temporal-difference learning algorithms*. University of Alberta, 2011.

[12] S. Mahadevan and B. Liu. Sparse Q-learning with mirror descent. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

[13] S. Mahadevan, B. Liu, P. S. Thomas, W. Dabney, S. Giguere, N. Jacek, I. Gemp, and J. L. 0002. Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces. *CoRR abs/1405.6757*, 2014.

[14] A. R. Mahmood and R. Sutton. Off-policy learning based on weighted importance sampling with linear computational complexity. In *Conference on Uncertainty in Artificial Intelligence*, 2015.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[16] R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 1998.

[17] R. Sutton, H. Maei, D. Precup, and S. Bhatnagar. Fast gradient-descent methods for temporal-difference learning with linear function approximation. *International Conference on Machine Learning*, 2009.

[18] R. Sutton, J. Modayil, M. Delp, T. Degris, P. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

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

[20] R. S. Sutton, A. R. Mahmood, D. Precup, and H. van Hasselt. A new Q(lambda) with interim forward view and Monte Carlo equivalence. *ICML*, 2014.

[21] R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2015.

[22] J. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997.

[23] H. van Hasselt, A. R. Mahmood, and R. Sutton. Off-policy TD (λ) with a true online equivalence. In *Conference on Uncertainty in Artificial Intelligence*, 2014.

[24] H. van Seijen and R. Sutton. True online TD(lambda). In *International Conference on Machine Learning*, 2014.

[25] C. Watkins. *Watkins: Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[26] A. White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

[27] H. Yu. On convergence of emphatic temporal-difference learning. In *Annual Conference on Learning Theory*, 2015.