Accelerated Gradient Temporal Difference Learning

Yangchen Pan, Adam White, Martha White
Department of Computer Science
Indiana University at Bloomington
{yangpan,adamw,martha}@indiana.edu

Abstract

The family of temporal difference (TD) methods span a spectrum from computationally frugal linear methods like TD(λ) to data efficient least squares methods. Least square methods make the best use of available data directly computing the TD solution and thus do not require tuning a typically highly sensitive learning rate parameter, but require quadratic computation and storage. Recent algorithmic developments have yielded several sub-quadratic methods that use an approximation to the least squares TD solution, but incur bias. In this paper, we propose a new family of accelerated gradient TD (ATD) methods that (1) provide similar data efficiency benefits to least-squares methods, at a fraction of the computation and storage (2) significantly reduce parameter sensitivity compared to linear TD methods, and (3) are asymptotically unbiased. We illustrate these claims with a proof of convergence in expectation and experiments on several benchmark domains and a large-scale industrial energy allocation domain.

Introduction

In reinforcement learning, a common strategy to learn an optimal policy is to iteratively estimate the value function for the current decision making policy—called *policy evaluation*—and then update the policy using the estimated values. The overall efficiency of this policy iteration scheme is directly influenced by the efficiency of the policy evaluation step. Temporal difference learning methods perform policy evaluation: they estimate the value function directly from the sequence of states, actions, and rewards produced by an agent interacting with an unknown environment.

The family of temporal difference methods span a spectrum from computationally-frugal, linear, stochastic approximation methods to data efficient but quadratic least squares TD methods. Stochastic approximation methods, such as temporal difference (TD) learning (Sutton 1988) and gradient TD methods (Maei 2011) perform approximate gradient descent on the mean squared projected Bellman error (MSPBE). These methods require linear (in the number of features) computation per time step and linear memory. These linear TD-based algorithms are well suited to problems with high dimensional feature vectors —compared to available resources— and domains where agent interaction occurs at a high rate (Szepesvari 2010). When the amount of data is limited or difficult to acquire, the feature vectors are small, or data efficiency is of primary concern, quadratic least squares TD (LSTD) methods may be preferred. These methods directly compute the value function that minimizes the MSPBE, and thus LSTD computes the same value function to which linear TD methods converge. Of course, there are many domains for which neither light weight linear TD methods, nor data efficient least squares methods may be a good match.

Significant effort has focused on reducing the computation and storage costs of least squares TD methods in order to span the gap between TD and LSTD. The iLSTD method (Geramifard and Bowling 2006) achieves sub-quadratic computation per time step, but still requires memory that is quadratic in the size of the features. The tLSTD method (Gehring, Pan, and White 2016) uses an incremental singular value decomposition (SVD) to achieve both sub-quadratic computation and storage. The basic idea is that in many domains the update matrix in LSTD can be replaced with a low rank approximation. In practice tLSTD achieves runtimes much closer to TD compared to iLSTD, while achieving better data efficiency. A related idea is to use random projections to reduce computation and storage of LSTD (Ghavamzadeh et al. 2010). In all these approaches, a scalar parameter (descent dimensions, rank, and number of projections), controls the balance between computation cost and quality of solution.

In this paper we explore a new approach called Accelerated gradient TD (ATD), that performs quasi-second-order gradient descent on the MSPBE. Our aim is to develop a family of algorithms that can interpolate between linear TD methods and LSTD, without incurring bias. ATD, when combined with a low-rank approximation, converges in expectation to the TD fixed-point, with convergence rate dependent on the choice of rank. Unlike previous subquadratic methods, consistency is guaranteed even when the rank is chosen to be one. We demonstrate the performance of ATD versus many linear and subquadratic methods in three domains, indicating that ATD (1) can match the data efficiency of LSTD, with significantly less computation and storage, (2) is unbiased, unlike many of the alternative subquadratic methods, (3) significantly reduces parameter sensitivity for the stepsize, versus linear TD methods, and (4) is significantly less sensitive to the choice of rank parameter than tLSTD, enabling a smaller rank to be chosen and so providing a more efficient incremental algorithm. Overall, the results suggest that ATD may be the first practical subquadratic complexity TD method suitable for fully incremental policy evaluation.

Background and Problem Formulation

In this paper we focus on the problem of *policy evaluation*, or that of learning a value function given a fixed policy. We model the interaction between an agent and its environment as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathrm{P}, r)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the set of actions, and $\mathrm{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ encodes the one-step state transition dynamics. On each discrete time step $t = 1, 2, 3, ...$, the agent selects an action according to its *behavior policy*, $A_t \sim \mu(S_t, \cdot)$, with $\mu : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ and the environment responds by transitioning into a new state S_{t+1} according to P, and emits a scalar reward $R_{t+1} \overset{\text{def}}{=} r(S_t, A_t, S_{t+1})$.

The objective under policy evaluation is to estimate the *value function*, $v_\pi : \mathcal{S} \to \mathbb{R}$, as the expected return from each state under some *target policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, \infty)$:

$$v_\pi(s) \overset{\text{def}}{=} \mathbb{E}_\pi[G_t | S_t = s],$$

where $\mathbb{E}_\pi$ denotes the expectation, defined over the future states encountered while selecting actions according to π. The *return*, denoted by $G_t \in \mathbb{R}$ is the discounted sum of future rewards given actions are selected according to π:

$$\begin{aligned} G_t &\overset{\text{def}}{=} R_{t+1} + \gamma_{t+1} R_{t+2} + \gamma_{t+1}\gamma_{t+2} R_{t+3} + ... \qquad (1) \\ &= R_{t+1} + \gamma_{t+1} G_{t+1} \end{aligned}$$

where $\gamma_{t+1} \in [0, 1]$ is a scalar that depends on S_t, A_t, S_{t+1} and discounts the contribution of future rewards exponentially with time. The generalization to transition-based discounting enables the unification of episodic and continuing tasks (White 2016) and so we adopt it here. In the standard continuing case, $\gamma_t = \gamma_c$ for some constant $\gamma_c < 1$ and for a standard episodic setting, $\gamma_t = 1$ until the end of an episode, at which point $\gamma_{t+1} = 0$, ending the infinite sum in the return. In the most common *on-policy* evaluation setting $\pi = \mu$, otherwise $\pi \neq \mu$ and policy evaluation problem is said to be *off-policy*.

In domains where the number of states is too large or the state is continuous, it is not feasible to learn the value of each state separately and we must generalize values between states using function approximation. In the case of linear function approximation the state is represented by fixed length feature vectors $x : \mathcal{S} \to \mathbb{R}^d$, where $\mathbf{x}_t \overset{\text{def}}{=} x(S_t)$ and the approximation to the value function is formed as a linear combination of a learned weight vector, $\mathbf{w} \in \mathbb{R}^d$, and $x(S_t)$: $v_\pi(S_t) \approx \mathbf{w}^\top \mathbf{x}_t$. The goal of policy evaluation is to learn $\mathbf{w}$ from samples generated while following μ.

The objective we pursue towards this goal is to minimize the mean-squared projected Bellman error (MSPBE):

$$\text{MSPBE}(\mathbf{w}, m) = (\mathbf{b}_m - \mathbf{A}_m \mathbf{w})^\top \mathbf{C}^{-1} (\mathbf{b}_m - \mathbf{A}_m \mathbf{w}) \qquad (2)$$

where $m : \mathcal{S} \to [0, \infty)$ is a weighting function,

$$\mathbf{A}_m \overset{\text{def}}{=} \mathbb{E}_\mu[(\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)^\top \mathbf{w}\, \mathbf{e}_{m,t}]$$

$$\mathbf{b}_m \overset{\text{def}}{=} \mathbb{E}_\mu[R_{t+1}\mathbf{e}_{m,t}]$$

$\mathbf{C}$ is any positive definite matrix, typically $\mathbf{C} = \mathbb{E}_\mu[\mathbf{x}_t\mathbf{x}_t^\top]$

with $\mathbf{b}_m - \mathbf{A}_m\mathbf{w} = \mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]$ for TD-error $\delta_t(\mathbf{w}) = R_{t+1} + \gamma_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w} - \mathbf{x}_t^\top\mathbf{w}$. The vector $\mathbf{e}_{m,t}$ is called the eligibility trace

$$\mathbf{e}_{m,t} \overset{\text{def}}{=} \rho_t(\gamma_{t+1}\lambda\mathbf{e}_{m,t-1} + M_t\mathbf{x}_t) \qquad \triangleright\ \rho_t \overset{\text{def}}{=} \frac{\pi(s_t, a_t)}{\mu(s_t, a_t)}$$

$$M_t \text{ s.t. } \mathbb{E}_\mu[M_t | S_t = s_t] = m(s)/d_\mu(s) \quad \text{if } d_\mu(s) \neq 0.$$

where $\lambda \in [0, 1]$ is called the trace-decay parameter and $d_\mu : \mathcal{S} \to [0, \infty)$ is the stationary distribution induced by following μ. The importance sampling ratio ρ_t reweights samples generated by μ to give an expectation over π

$$\begin{aligned} &\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}] \\ &\quad = \sum_{s\in\mathcal{S}} d_\mu(s)\mathbb{E}_\pi[\delta_t(\mathbf{w})(\gamma\lambda\mathbf{e}_{m,t-1} + M_t\mathbf{x}_t) | S_t = s]. \end{aligned}$$

This re-weighting enables v_π to be learned from samples generated by μ (under off-policy sampling).

The most well-studied weighting occurs when $M_t = 1$ (i.e., $m(s) = d_\mu(s)$). In the on-policy setting, with $\mu = \pi$, $\rho_t = 1$ for all t and $m(s) = d_\pi(s)$ the $\mathbf{w}$ that minimizes the MSPBE is the same as the $\mathbf{w}$ found by the on-policy temporal difference learning algorithm called TD(λ). More recently, a new *emphatic* weighting was introduced with the emphatic TD (ETD) algorithm, which we denote m_{ETD}. This weighting includes long-term information about π (see (Sutton, Mahmood, and White 2016, Pg. 16)),

$$M_t = \lambda_t + (1 - \lambda_t)F_t \qquad \triangleright\ F_t = \gamma_t\rho_{t-1}F_{t-1} + 1.$$

Importantly, the $\mathbf{A}_{m_{\text{ETD}}}$ matrix induced by the emphatic weighting is positive semi-definite (Yu 2015; Sutton, Mahmood, and White 2016), which we will later use to ensure convergence of our algorithm under both on- and off-policy sampling. The $\mathbf{A}_{d_\mu}$ used by TD(λ) is not necessarily positive semi-definite, and so TD(λ) can diverge when $\pi \neq \mu$ (off-policy).

Two common strategies to obtain the minimum $\mathbf{w}$ of this objective are stochastic temporal difference techniques, such as TD(λ) (Sutton 1988), or directly approximating the linear system and solving for the weights, such as in LSTD(λ) (Boyan 1999). The first class constitute linear complexity methods, both in computation and storage, including the family of gradient TD methods (Maei 2011), true online TD methods (van Seijen and Sutton 2014; van Hasselt, Mahmood, and Sutton 2014) and several others (see (Dann, Neumann, and Peters 2014; White and White 2016) for a more complete summary). On the other extreme, with quadratic computation and storage, one can approximate $\mathbf{A}_m$ and $\mathbf{b}_m$ incrementally and solve the system $\mathbf{A}_m\mathbf{w} = \mathbf{b}_m$. Given a batch of t samples $\{(S_i, A_i, S_{i+1}, R_{i+1})\}_{i=1}^t$, one can estimate

$$\mathbf{A}_{m,t} \overset{\text{def}}{=} \frac{1}{t}\sum_{i=1}^{t} \mathbf{e}_{m,i}(\mathbf{x}_i - \gamma\mathbf{x}_{i+1})^\top$$

$$\mathbf{b}_{m,t} \overset{\text{def}}{=} \frac{1}{t}\sum_{i=1}^{t} \mathbf{e}_{m,i}R_{i+1},$$

and then compute solution $\mathbf{w}$ such that $\mathbf{A}_{m,t}\mathbf{w} = \mathbf{b}_{m,t}$. Least-squares TD methods are typically implemented incrementally using the Sherman-Morrison formula, requiring $\mathcal{O}(d^2)$ storage and computation per step.

Our goal is to develop algorithms that interpolate between these two extremes, which we discuss in the next section.

Algorithm derivation

To derive the new algorithm, we first take the gradient of the MSPBE (in 2) to get

$$-\frac{1}{2}\nabla_{\mathbf{w}}\text{MSPBE}(\mathbf{w}, m) = \mathbf{A}_m^\top \mathbf{C}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]. \quad (3)$$

Consider a second order update by computing the Hessian: $\mathbf{H} = \mathbf{A}_m^\top \mathbf{C}^{-1}\mathbf{A}_m^\top$. For simplicity of notation, let $\mathbf{A} = \mathbf{A}_m$ and $\mathbf{b} = \mathbf{b}_m$. For invertible $\mathbf{A}$, the second-order update is

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \tfrac{\alpha_t}{2}\mathbf{H}^{-1}\nabla_{\mathbf{w}}\text{MSPBE}(\mathbf{w}, m)\\ &= \mathbf{w}_t + \alpha_t(\mathbf{A}^\top\mathbf{C}^{-1}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{C}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]\\ &= \mathbf{w}_t + \alpha_t\mathbf{A}^{-1}\mathbf{C}\mathbf{A}^{-\top}\mathbf{A}^\top\mathbf{C}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]\\ &= \mathbf{w}_t + \alpha_t\mathbf{A}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]\end{aligned}$$

In fact, for our quadratic loss, the optimal descent direction is $\mathbf{A}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]$ with $\alpha_t = 1$, in the sense that $\text{argmin}_{\Delta\mathbf{w}}\,\text{loss}(\mathbf{w}_t + \Delta\mathbf{w}) = \mathbf{A}^{-1}\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]$. Computing the Hessian and updating $\mathbf{w}$ requires quadratic computation, and in practice quasi-Newton approaches are used that approximate the Hessian. Additionally, there have been recent insights that using approximate Hessians for stochastic gradient descent can in fact speed convergence (Schraudolph, Yu, and Günter 2007; Bordes, Bottou, and Gallinari 2009; Mokhtari and Ribeiro 2014). These methods maintain an approximation to the Hessian, and sample the gradient. This Hessian approximation provides curvature information that can significantly speed convergence, as well as reduce parameter sensitivity to the stepsize.

Our objective is to improve on the sample efficiency of linear TD methods, while avoiding both quadratic computation and asymptotic bias. First, we need an approximation $\hat{\mathbf{A}}$ to $\mathbf{A}$ that provides useful curvature information, but that is also sub-quadratic in storage and computation. Second, we need to ensure that the approximation, $\hat{\mathbf{A}}$, does not lead to a biased solution $\mathbf{w}$.

We propose to achieve this by approximating only $\mathbf{A}^{-1}$ and sampling $\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}] = \mathbf{b} - \mathbf{A}\mathbf{w}$ using $\delta_t(\mathbf{w}_t)\mathbf{e}_t$ as an unbiased sample. The proposed accelerated temporal difference learning update—which we call ATD(λ)—is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (\alpha_t\hat{\mathbf{A}}_t^\dagger + \eta\mathbf{I})\delta_t\mathbf{e}_t$$

with expected update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (\alpha_t\hat{\mathbf{A}}^\dagger + \eta\mathbf{I})\mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}] \quad (4)$$

with regularization $\eta > 0$. If $\hat{\mathbf{A}}$ is a poor approximation of $\mathbf{A}$, or discards key information—as we will do with a low rank approximation— then updating using only $\mathbf{b} - \hat{\mathbf{A}}\mathbf{w}$ will result in a biased solution, as is the case for tLSTD (Gehring, Pan, and White 2016, Theorem 1). Instead, sampling $\mathbf{b} - \mathbf{A}\mathbf{w} = \mathbb{E}_\mu[\delta_t(\mathbf{w})\mathbf{e}_{m,t}]$, as we show in Theorem 1, yields an unbiased solution, even with a poor approximation $\hat{\mathbf{A}}$. The regularization $\eta > 0$ is key to ensure this consistency, by providing a full rank preconditioner $\alpha_t\hat{\mathbf{A}}_t^\dagger + \eta\mathbf{I}$.

Given the general form of ATD(λ), the next question is how to approximate $\mathbf{A}$. Two natural choices are a diagonal approximation and a low-rank approximation. Storing and using a diagonal approximation would only require linear $O(d)$ time and space. For a low-rank approximation $\hat{\mathbf{A}}$, of rank k, represented with truncated singular value decomposition $\hat{\mathbf{A}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$, the storage requirement is $O(dk)$ and the required matrix-vector multiplications are only $O(dk)$ because for any vector $\mathbf{v}$, $\hat{\mathbf{A}}\mathbf{v} = \mathbf{U}_k\mathbf{\Sigma}_k(\mathbf{V}_k^\top\mathbf{v})$, is a sequence of $O(dk)$ matrix-vector multiplications. Exploratory experiments revealed that the low-rank approximation approach significantly outperformed the diagonal approximation. In general, however, many other approximations to $\mathbf{A}$ could be used, which is an important direction for ATD.

We opt for an incremental SVD, that previously proved effective for incremental estimation in reinforcement learning (Gehring, Pan, and White 2016). The total computational complexity of the algorithm is $\mathcal{O}(dk + k^3)$ for the fully incremental update to $\hat{\mathbf{A}}$ and $\mathcal{O}(dk)$ for mini-batch updates of k samples. Notice that when $k = 0$, the algorithm reduces exactly to TD(λ), where η is the stepsize. On the other extreme, where $k = d$, ATD is equivalent to an iterative form of LSTD(λ). See the appendix for a further discussion, and implementation details.

Convergence of ATD(λ)

As with previous convergence results for temporal difference learning algorithms, the first key step is to prove that the expected update converges to the TD fixed point. Unlike previous proofs of convergence in expectation, we do not require the true $\mathbf{A}$ to be full rank. This generalization is important, because as shown previously, $\mathbf{A}$ is often low-rank, even if features are linearly independent (Bertsekas 2007; Gehring, Pan, and White 2016). Further, ATD should be more effective if $\mathbf{A}$ is low-rank, and so requiring a full-rank $\mathbf{A}$ would limit the typical use-cases for ATD.

To get across the main idea, we first prove convergence of ATD with weightings that give positive semi-definite $\mathbf{A}_m$; a more general proof for other weightings is in the appendix.

Assumption 1. $\mathbf{A}$ *is diagonalizable, that is, there exists invertible* $\mathbf{Q} \in \mathbb{R}^{d\times d}$ *with normalized columns (eigenvectors) and diagonal* $\mathbf{\Lambda} \in \mathbb{R}^{d\times d}$, $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_d)$, *such that* $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$. *Assume the ordering* $\lambda_1 \geq \ldots \geq \lambda_d$.

Assumption 2. $\alpha \in (0, 2)$ *and* $0 < \eta \leq \lambda_1^{-1}\max(2-\alpha, \alpha)$.

Finally, we introduce an assumption that is only used to characterize the convergence rate. This condition has been previously used (Hansen 1990; Gehring, Pan, and White 2016) to enforce a level of smoothness on the system.

Assumption 3. *The linear system defined by* $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$ *and* $\mathbf{b}$ *satisfy the discrete Picard condition: for some* $p > 1$, $|(\mathbf{Q}^{-1}\mathbf{b})_j| \leq \lambda_j^p$ *for all* $j = 1, \ldots,$ *rank*$(\mathbf{A})$.

Theorem 1. *Under Assumptions 1 and 2, for any $k \geq 0$, let $\hat{\mathbf{A}}$ be the rank-k approximation $\hat{\mathbf{A}} = \mathbf{Q}\mathbf{\Lambda}_k\mathbf{Q}^{-1}$ of $\mathbf{A}_m$, where $\mathbf{\Lambda}_k \in \mathbb{R}^{d\times d}$ with $\mathbf{\Lambda}_k(j,j) = \lambda_j$ for $j = 1,\ldots,k$ and zero otherwise. If $m = d_\mu$ or m_{ETD}, the expected updating rule in* (4) *converges to the fixed-point $\mathbf{w}^\star = \mathbf{A}_m^\dagger \mathbf{b}_m$. Further, if Assumption 3 is satisfied, the convergence rate is*

$$\|\mathbf{w}_t - \mathbf{w}^\star\| \leq \max\Bigg(\max_{j\in\{1,\ldots,k\}} |1-\alpha-\eta\lambda_j|^t \lambda_j^{p-1}, \max_{j\in\{k+1,\ldots,\textit{rank}(\mathbf{A})\}} |1-\eta\lambda_j|^t\lambda_j^{p-1}\Bigg)$$

Proof: We use a general result about stationary iterative methods which is applicable to the case where $\mathbf{A}$ is not full rank. Theorem 1.1 (Shi, Wei, and Zhang 2011) states that given a singular and consistent linear system $\mathbf{A}\mathbf{w} = \mathbf{b}$ where $\mathbf{b}$ is in the range of $\mathbf{A}$, the stationary iteration with $\mathbf{w}_0 \in \mathbb{R}^d$ for $t = 1, 2, \ldots$

$$\mathbf{w}_t = (\mathbf{I} - \mathbf{B}\mathbf{A})\mathbf{w}_{t-1} + \mathbf{B}\mathbf{b} \tag{5}$$

converges to the solution $\mathbf{w} = \mathbf{A}^\dagger\mathbf{b}$ if and only if the following three conditions are satisfied.

Condition **I**: the eigenvalues of $\mathbf{I} - \mathbf{B}\mathbf{A}$ are equal to 1 or have absolute value strictly less than 1.

Condition **II**: $\text{rank}(\mathbf{B}\mathbf{A}) = \text{rank}[(\mathbf{B}\mathbf{A})^2]$.

Condition **III**: $\text{nullspace}(\mathbf{B}\mathbf{A}) = \text{nullspace}(\mathbf{A})$.

We verify these conditions to prove the result. First, because we use the projected Bellman error, $\mathbf{b}$ is in the range of $\mathbf{A}$ and the system is consistent: there exists $\mathbf{w}$ s.t. $\mathbf{A}\mathbf{w} = \mathbf{b}$.

To rewrite our updating rule (4) to be expressible in terms of (5), let $\mathbf{B} = \alpha\hat{\mathbf{A}}^\dagger + \eta\mathbf{I}$, giving

$$\begin{aligned}\mathbf{B}\mathbf{A} &= \alpha\hat{\mathbf{A}}^\dagger\mathbf{A} + \eta\mathbf{A} = \alpha\mathbf{Q}\mathbf{\Lambda}_k^\dagger\mathbf{Q}^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} + \eta\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}\\ &= \alpha\mathbf{Q}\mathbf{I}_k\mathbf{Q}^{-1} + \eta\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}\\ &= \mathbf{Q}(\alpha\mathbf{I}_k + \eta\mathbf{\Lambda})\mathbf{Q}^{-1} \end{aligned}\tag{6}$$

where $\mathbf{I}_k$ is a diagonal matrix with the indices $1,\ldots,k$ set to 1, and the rest zero.

Proof for condition I. Using (6), $\mathbf{I} - \mathbf{B}\mathbf{A} = \mathbf{Q}(\mathbf{I} - \alpha\mathbf{I}_k - \eta\mathbf{\Lambda})\mathbf{Q}^{-1}$. To bound the maximum absolute value in the diagonal matrix $\mathbf{I} - \alpha\mathbf{I}_k - \eta\mathbf{\Lambda}$, we consider eigenvalue λ_j in $\mathbf{\Lambda}$, and address two cases. Because $\mathbf{A}_m$ is positive semi-definite for the assumed m (Sutton, Mahmood, and White 2016), $\lambda_j \geq 0$ for all $j = 1,\ldots,d$.

Case 1: $j \leq k$.

$$\begin{aligned}&|1-\alpha-\eta\lambda_j| && \triangleright \text{ for } 0 < \eta < \max\left(\frac{2-\alpha}{\lambda_1}, \frac{\alpha}{\lambda_1}\right)\\ &< \max(|1-\alpha|, |1-\alpha-(2-\alpha)|, |1-\alpha-\alpha|)\\ &= \max(|1-\alpha|, 1, 1) < 1 && \triangleright \text{ because } \alpha \in (0,2).\end{aligned}$$

Case 2: $j > k$. $|1-\eta\lambda_j| < 1$ if $0 < \eta < 2/\lambda_j$ which is true for $\eta = \lambda_1^{-1}\max(2-\alpha,\alpha)$ for any $\alpha \in (0,2)$.

Proof for condition II. $(\mathbf{B}\mathbf{A})^2$ does not change the number of positive eigenvalues, so the rank is unchanged.

Proof for condition III. To show the nullspaces of $\mathbf{B}\mathbf{A}$ and $\mathbf{A}$ are equal, it is sufficient to prove $\mathbf{B}\mathbf{A}\mathbf{w} = \mathbf{0}$ if and only if $\mathbf{A}\mathbf{w} = \mathbf{0}$. $\mathbf{B} = \mathbf{Q}(\alpha\mathbf{\Lambda}_k + \eta\mathbf{I})\mathbf{Q}^{-1}$, is invertible because $\eta > 0$ and $\lambda_j \geq 0$. For any $\mathbf{w} \in \text{nullspace}(\mathbf{A})$, we get $\mathbf{B}\mathbf{A}\mathbf{w} = \mathbf{B}\mathbf{0} = \mathbf{0}$, and so $\mathbf{w} \in \text{nullspace}(\mathbf{B}\mathbf{A})$. For any $\mathbf{w} \in \text{nullspace}(\mathbf{B}\mathbf{A})$, $\mathbf{B}\mathbf{A}\mathbf{w} = \mathbf{0} \implies \mathbf{A}\mathbf{w} = \mathbf{B}^{-1}\mathbf{0} = \mathbf{0}$, and so $\mathbf{w} \in \text{nullspace}(\mathbf{A})$.

Convergence rate. Assume $\mathbf{w}_0 = \mathbf{0}$. On each step, we update with $\mathbf{w}_{t+1} = (\mathbf{I}-\mathbf{B}\mathbf{A})\mathbf{w}_t + \mathbf{B}\mathbf{b} = \sum_{i=0}^{t-1}(\mathbf{I}-\mathbf{B}\mathbf{A})^i\mathbf{B}\mathbf{b}$. This can be verified inductively, where

$$\begin{aligned}\mathbf{w}_{t+1} &= (\mathbf{I}-\mathbf{B}\mathbf{A})\sum_{i=0}^{t-2}(\mathbf{I}-\mathbf{B}\mathbf{A})^i\mathbf{B}\mathbf{b} + (\mathbf{I}-\mathbf{B}\mathbf{A})^0\mathbf{B}\mathbf{b}\\ &= \sum_{i=0}^{t-1}(\mathbf{I}-\mathbf{B}\mathbf{A})^i\mathbf{B}\mathbf{b}.\end{aligned}$$

For $\bar{\mathbf{\Lambda}} = \mathbf{I} - \alpha\mathbf{I}_k - \eta\mathbf{\Lambda}$, because $(\mathbf{I}-\mathbf{B}\mathbf{A})^i = \mathbf{Q}\bar{\mathbf{\Lambda}}^i\mathbf{Q}^{-1}$,

$$\begin{aligned}\mathbf{w}_t &= \mathbf{Q}\left(\sum_{i=0}^{t-1}\bar{\mathbf{\Lambda}}^i\right)\mathbf{Q}^{-1}\mathbf{Q}(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})\mathbf{Q}^{-1}\mathbf{b}\\ &= \mathbf{Q}\left(\sum_{i=0}^{t-1}\bar{\mathbf{\Lambda}}^i\right)(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})\mathbf{Q}^{-1}\mathbf{b}\end{aligned}$$

and because $\mathbf{w}_t \to \mathbf{w}^\star$,

$$\begin{aligned}\|\mathbf{w}_t - \mathbf{w}^\star\| &= \|\mathbf{Q}\left(\sum_{i=0}^{\infty}\bar{\mathbf{\Lambda}}^i - \sum_{i=0}^{t}\bar{\mathbf{\Lambda}}^i\right)(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})\mathbf{Q}^{-1}\mathbf{b}\|\\ &= \|\mathbf{Q}\bar{\mathbf{\Lambda}}_t(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})\mathbf{Q}^{-1}\mathbf{b}\| \qquad \triangleright\ \bar{\mathbf{\Lambda}}_t(j,j) \overset{\text{def}}{=} \frac{\bar{\lambda}_j^t}{1-\bar{\lambda}_j}\\ &\leq \|\mathbf{Q}\|\|\bar{\mathbf{\Lambda}}_t(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})\mathbf{Q}^{-1}\mathbf{b}\|\end{aligned}$$

where $\|\mathbf{Q}\| \leq 1$ because $\mathbf{Q}$ has normalized columns.

For $j = 1,\ldots,k$, we have that the magnitude of the values in $\bar{\mathbf{\Lambda}}_t(\alpha\mathbf{\Lambda}_k^\dagger + \eta\mathbf{I})$ are

$$\frac{(1-\alpha-\eta\lambda_j)^t}{\alpha+\eta\lambda_j}(\alpha\lambda_j^{-1} + \eta) = \frac{(1-\alpha-\eta\lambda_j)^t}{\lambda_j}.$$

For $j = k,\ldots,\text{rank}(\mathbf{A})$, we get $\frac{(1-\eta\lambda_j)^t}{\lambda_j}$.

Under the discrete Picard condition, $|(\mathbf{Q}^{-1}\mathbf{b})_j| \leq \lambda_j^p$ and so the denominator λ_j cancels, giving the desired result. ■

This theorem gives insight into the utility of ATD for speeding convergence, as well as the effect of k. Consider TD(λ), which has positive definite $\mathbf{A}$ in on-policy learning (Sutton 1988, Theorem 2). The above theorem guarantees ATD convergences to the TD fixed-point, for any k. For $k = 0$, the expected ATD update is exactly the expected TD update. Now, we can compare the convergence rate of TD and ATD, using the above convergence rate.

Take for instance the setting $\alpha = 1$ for ATD, which is common for second-order methods and let $p = 2$. The rate of convergence reduces to the maximum of $\max_{j\in\{1,\ldots,k\}}\eta^t\lambda_j^{t+1}$ and $\max_{j\in\{k+1,\ldots,\text{rank}(\mathbf{A})\}}|1-\eta\lambda_j|^t\lambda_j$. In early learning, the convergence rate for TD is dominated by $|1-\eta\lambda_1|^t\lambda_1$, because λ_j is largest relative to $|1-\eta\lambda_j|^t$ for small t. ATD, on the other hand, for a larger k, can pick a smaller η and so has a much smaller value for $j = 1$, i.e., $\eta^t \lambda_1^{t+1}$, and $|1-\eta\lambda_j|^t \lambda_j$ is small because λ_j is small for $j > k$. As k gets smaller, $|1 - \eta\lambda_{k+1}|^t \lambda_{k+1}$ becomes larger, slowing convergence. For low-rank domains, however, k could be quite small and the preconditioner could still improve the convergence rate in early learning—potentially significantly outperforming TD.

ATD is a quasi-second order method, meaning sensitivity to parameters should be reduced and thus it should be simpler to set the parameters. The convergence rate provides intuition that, for reasonably chosen k, the regularizer η should be small—smaller than a typical stepsize for TD. Additionally, because ATD is a stochastic update, not the expected update, we make use of typical conventions from stochastic gradient descent to set our parameters. We set $\alpha_t = \frac{\alpha_0}{t}$, as in previous stochastic second-order methods (Schraudolph, Yu, and Günter 2007), where we choose $\alpha_0 = 1$ and set η to a small fixed value. Our choice for η represents a small final stepsize, as well as matching the convergence rate intuition.

On the bias of subquadratic methods. The ATD(λ) update was derived to ensure convergence to the minimum of the MSPBE, either for the on-policy or off-policy setting. Our algorithm summarizes past information, in $\hat{\mathbf{A}}$, to improve the convergence rate, without requiring quadratic computation and storage. Prior work aspired to the same goal, however, the resultant algorithms are biased. The iLSTD algorithm can be shown to converge for a specific class of feature selection mechanisms (Geramifard, Bowling, and Zinkevich 2007, Theorem 2); this class, however, does not include the greedy mechanism that is used in iLSTD algorithm to select a descent direction. The random projections variant of LSTD (Ghavamzadeh et al. 2010) can significantly reduce the computational complexity compared with conventional LSTD, with projections down to size k, but the reduction comes at a cost of an increase in the approximation error (Ghavamzadeh et al. 2010). Fast LSTD (Prashanth, Korda, and Munos 2013) does randomized TD updates on a batch of data; this algorithm could be run incrementally with O(dk) by using mini-batches of size k. Though it has a nice theoretical characterization, this algorithm is restricted to $\lambda = 0$. Finally, the most related algorithm is tLSTD, which also uses a low-rank approximation to $\mathbf{A}$.

In ATD $\hat{\mathbf{A}}_t$ is used very differently, from how $\hat{\mathbf{A}}_t$ is used in tLSTD. The tLSTD algorithm uses a similar approximation $\hat{\mathbf{A}}_t$ as ATD, but tLSTD uses it to compute a closed form solution $\mathbf{w}_t = \hat{\mathbf{A}}_t^\dagger \mathbf{b}_t$, and thus is biased (Gehring, Pan, and White 2016, Theorem 1). In fact, the bias grows with decreasing k, proportionally to the magnitude of the kth largest singular value of $\mathbf{A}$. In ATD, the choice of k is decoupled from the fixed point, and so can be set to balance learning speed and computation with no fear of asymptotic bias.

Empirical Results

All the following experiments investigate the on-policy setting, and thus we make use of the standard version of ATD for simplicity. Future work will explore off-policy domains with the emphatic update. The results presented in this section were generated over 756 thousand individual experiments run on three different domains. Due to space constraints detailed descriptions of each domain, error calculation, and all other parameter settings are discussed in detail in the appendix. We included a wide variety of baselines in our experiments, additional related baselines excluded from our study are also discussed in the appendix.

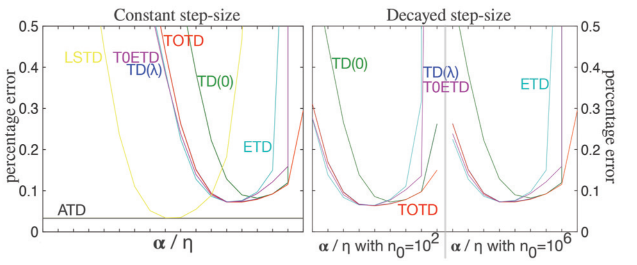

Figure 1: Parameter sensitivity in Boyan's chain with constant stepsize (LHS) and decayed stepsizes (RHS). In the plots above, each point summarizes the mean performance (over 1000 time steps) of an algorithm for one setting of α for linear methods, or η for LSTD, and $\alpha/100$ regularizer for ATD, using percentage error compared to the true value function. In the decayed stepsize case, where $\alpha_t = \alpha_0 \frac{n_0+1}{n_0+\text{episode\#}}$, 18 values of α_0 and two values of n_0 were tested—corresponding to the two sides of the RHS graph. The LSTD algorithm (in yellow) has no parameters to decay. Our ATD algorithm (in black) achieves the lowest error in this domain, and exhibits little sensitivity to it's regularization parameter (with stepsize as $\alpha_t = \frac{1}{t}$ across all experiments).

Our first batch of experiments were conducted on Boyan's chain—a domain known to elicit the strong advantages of LSTD(λ) over TD(λ). In Boyan's chain the agent's objective is to estimate the value function based on a low-dimensional, dense representation of the underlying state (perfect representation of the value function is possible). The ambition of this experiment was to investigate the performance of ATD in a domain where the pre-conditioner matrix is full rank; no rank truncation is applied. We compared five linear-complexity methods (TD(0), TD(λ), true online TD(λ), ETD(λ), true online ETD(λ)), against LSTD(λ) and ATD, reporting the percentage error relative to the true value function over the first 1000 steps, averaged over 200 independent runs. We swept a large range of stepsize parameters, trace decay rates, and regularization parameters, and tested both fixed and decaying stepsize schedules. Figure 1 summarizes the results.

Both LSTD(λ) and ATD achieve lower error compared to all the linear baselines—even thought each linear method was tuned using 864 combinations of stepsizes and λ. In terms of sensitivity, the choice of stepsize for TD(0) and ETD exhibit large effect on performance (indicated by sharp valleys), whereas true-online TD(λ) is the least sensitive to learning rate. LSTD(λ) using the Sherman-Morrison update (used in many prior empirical studies) is sensitive to the regularization parameter; the parameter free nature of LSTD may be slightly overstated in the literature.[1]

[1]We are not the first to observe this. Sutton and Barto (2016)

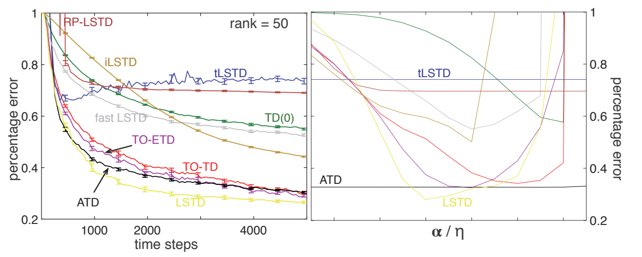

Figure 2: The learning curves (LHS) are percentage error versus time steps averaged over 100 runs of ATD with rank 50, LSTD and several baselines described in text. The sensitivity plot (RHS) is with respect to the learning rate of the linear methods, and regularization parameter of the matrix methods. The tLSTD algorithm has no parameter besides rank, while ATD has little sensitivity to it's regularization parameter.

Our second batch of experiments investigated characteristics of ATD in a classic benchmark domain with a sparse high-dimensional feature representation where perfect approximation of the value function is not possible—Mountain car with tile coding. The policy to be evaluated stochastically takes the action in the direction of the sign of the velocity, with performance measured by computing a truncated Monte Carlo estimate of the return from states sampled from the stationary distribution (detailed in the appendix). We used a fine grain tile coding of the the 2D state, resulting in a 1024 dimensional feature representation with exactly 10 units active on every time step. We tested TD(0), true online TD(λ) true online ETD(λ), and sub-quadratic methods, including iLSTD, tLSTD, random projection LSTD, and fast LSTD (Prashanth, Korda, and Munos 2013). As before a wide range of parameters (α, λ, η) were swept over a large set. Performance was averaged over 100 independent runs. A fixed stepsize schedule was used for the linear TD baselines, because that achieved the best performance. The results are summarized in figure 2.

LSTD and ATD exhibit faster initial learning compared to all other methods. This is particularly impressive since k is less than 5% of the size of $\mathbf{A}$. Both fast LSTD and projected LSTD perform considerably worse than the linear TD-methods, while iLSTD exhibits high parameter sensitivity. tLSTD has no tunable parameter besides k, but performs poorly due to the high stochasticity in the policy—additional experiments with randomness in action selection of 0% and 10% yielded better performance for tLSTD, but never equal to ATD. The true online linear methods perform very well compared to ATD, but this required sweeping hundreds of combinations of α and λ, whereas ATD exhibited little sensitivity to it's regularization parameter (see Figure 2 RHS); ATD achieved excellent performance with the same parameter setting as we used in Boyan's chain.[2]

note that η plays a role similar to the stepsize for LSTD.

[2]For the remaining experiments in the paper, we excluded the TD methods without true online traces because they perform worse than their true online counterparts in all our experiments. This result matches the results in the literature (van Seijen et al. 2016).

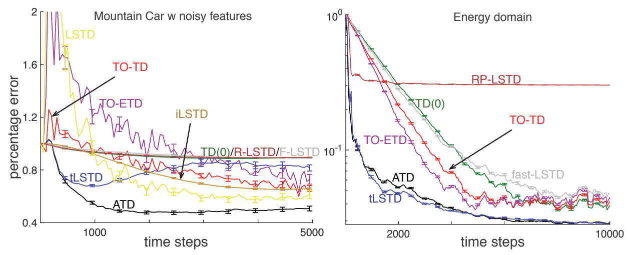

Figure 3: Learning curves on Mountain Car with noisy features (LHS) and on Energy allocation (RHS), in log scale.

We ran an additional experiment in Mountain car to more clearly exhibit the benefit of ATD over existing methods. We used the same setting as above, except that 100 additional features were added to the feature vector, with 50 of them randomly set to one and the rest zero. This noisy feature vector is meant to emulate a situation such as a robot that has a sensor that becomes unreliable, generating noisy data, but the remaining sensors are still useful for the task at hand. The results are summarized in Figure 3. Naturally all methods are adversely effected by this change, however ATD's low rank approximation enables the agent to ignore the unreliable feature information and learn efficiently. tLSTD, as suggested by our previous experiments does not seem to cope well with the increase in stochasticity.

Our final experiment compares the performance of several sub-quadratic complexity policy evaluation methods in an industrial energy allocation simulator with much larger feature dimension (see Figure 3). As before we report percentage error computed from Monte Carlo rollouts, averaging performance over 50 independent runs and selecting and testing parameters from an extensive set (detailed in the appendix). The policy was optimized ahead of time and fixed, and the feature vectors were produced via tile coding, resulting in an 8192 dimensional feature vector with 800 units active on each step. Although the feature dimension here is still relatively small, a quadratic method like LSTD nonetheless would require over 67 million operations per time step, and thus methods that can exploit low rank approximations are of particular interest. The results indicate that both ATD and tLSTD achieve the fastest learning, as expected. The intrinsic rank in this domain appears to be small compared to the feature dimension—which is exploited by ATD and tLSTD with $r = 40$—while the performance of tLSTD indicates that the domain exhibits little stochasticity. The appendix contains additional results for this domain—in the small rank setting ATD significantly outperforms tLSTD.

Conclusion and future work

In this paper, we introduced a new family of TD learning algorithms that take a fundamentally different approach from previous incremental TD algorithms. The key idea is to use a preconditioner on the temporal difference update, similar to a quasi-Newton stochastic gradient descent update. We prove that the expected update is consistent, and empirically demonstrated improved learning speed and parameter insensitivity, even with significant approximations in the preconditioner.

This paper only begins to scratch the surface of potential preconditioners for ATD. There remains many avenues to explore the utility of other preconditioners, such as diagonal approximations, eigenvalues estimates, other matrix factorizations and approximations to $\mathbf{A}$ that are amenable to inversion. The family of ATD algorithms provides a promising avenue for more effectively using results for stochastic gradient descent to improve sample complexity, with feasible computational complexity.

Acknowledgements This material[3] is based upon work supported by the National Science Foundation under grant no. IIS-1566186.

References

Bertsekas, D. 2007. *Dynamic Programming and Optimal Control*. Athena Scientific Press.

Bordes, A.; Bottou, L.; and Gallinari, P. 2009. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*.

Boyan, J. A. 1999. Least-squares temporal difference learning. *International Conference on Machine Learning*.

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

Gehring, C.; Pan, Y.; and White, M. 2016. Incremental Truncated LSTD. In *International Joint Conference on Artificial Intelligence*.

Geramifard, A., and Bowling, M. 2006. Incremental least-squares temporal difference learning. In *AAAI Conference on Artificial Intelligence*.

Geramifard, A.; Bowling, M.; and Zinkevich, M. 2007. iLSTD: Eligibility traces and convergence analysis. In *Advances in Neural Information Processing Systems*.

Ghavamzadeh, M.; Lazaric, A.; Maillard, O. A.; and Munos, R. 2010. LSTD with random projections. In *Advances in Neural Information Processing Systems*.

Hansen, P. C. 1990. The discrete picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*.

Maei, H. 2011. *Gradient Temporal-Difference Learning Algorithms*. Ph.D. Dissertation, University of Alberta.

Mokhtari, A., and Ribeiro, A. 2014. RES: Regularized stochastic BFGS algorithm. *IEEE Transactions on Signal Processing*.

Prashanth, L. A.; Korda, N.; and Munos, R. 2013. Fast LSTD using stochastic approximation: Finite time analysis and application to traffic control. *ECML PKDD*.

Schraudolph, N.; Yu, J.; and Günter, S. 2007. A stochastic quasi-Newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*.

Shi, X.; Wei, Y.; and Zhang, W. 2011. Convergence of general nonstationary iterative methods for solving singular linear equations. *SIAM Journal on Matrix Analysis and Applications*.

Sutton, R., and Barto, A. G. 2016. *Reinforcement Learning: An Introduction 2nd Edition*. MIT press.

Sutton, R. S.; Mahmood, A. R.; and White, M. 2016. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*.

Szepesvari, C. 2010. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.

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

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

van Seijen, H.; Mahmood, R. A.; Pilarski, P. M.; Machado, M. C.; and Sutton, R. S. 2016. True Online Temporal-Difference Learning. In *Journal of Machine Learning Research*.

White, A. M., and White, M. 2016. Investigating practical, linear temporal difference learning. In *International Conference on Autonomous Agents and Multiagent Systems*.

White, M. 2016. Unifying task specification in reinforcement learning. *arXiv.org*.

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

[3]The full version of the paper, including the appendix, can be found on arXiv.