Accelerated Gradient Temporal Difference Learning

Yangchen Pan, Adam White, and Martha White
{YANGPAN,ADAMW,MARTHA}@INDIANA.EDU
Department of Computer Science, Indiana University

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 squares 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.

1. Introduction

Temporal difference learning encompasses both 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. LSTD directly computes the value function that minimizes the MSPBE, and thus 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 et al., 2016) uses an incremental singular value decomposition (SVD) to achieve both sub-quadratic computation and storage. 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), helps the user tradeoff computational efficiency and bias.

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 described above, 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, (3) significantly reduces parameter sensitivity to step-size choice, compared with 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.

2. 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 $R_{t+1} \stackrel{\text{def}}{=} r(S_t, A_t, S_{t+1})$.

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* $\pi : \mathcal{S} \times \mathcal{A} \to [0, \infty)$: $v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t | S_t = s]$, where $\mathbb{E}_\pi$ denotes the expectation, defined over the future states encountered while selecting actions according to π. The *return* $G_t \in \mathbb{R}$, is the discounted sum of future rewards given actions are selected according to π:

$$G_t \stackrel{\text{def}}{=} R_{t+1} + \gamma_{t+1} R_{t+2} + \gamma_{t+1}\gamma_{t+2} R_{t+3} + ... = R_{t+1} + \gamma_{t+1} G_{t+1}$$

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 $\gamma_{t+1} = 0$, truncating the infinite sum in the return. In the most common *on-policy* evaluation setting $\pi = \mu$, otherwise $\pi \neq \mu$ and 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 $\mathbf{x}_t \stackrel{\text{def}}{=} x(S_t)$ 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):

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

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

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

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

$$\mathbf{e}_{m,t} \stackrel{\text{def}}{=} \rho_t(\gamma_{t+1}\lambda\mathbf{e}_{m,t-1} + M_t\mathbf{x}_t), \qquad M_t \text{ s.t. } \mathbb{E}_\mu[M_t|S_t = s_t] = m(s)/d_\mu(s) \quad \text{if } d_\mu(s) \neq 0.$$

where $\rho_t \stackrel{\text{def}}{=} \frac{\pi(s_t,a_t)}{\mu(s_t,a_t)}$, $\lambda \in [0,1]$ 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

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

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 et al., 2016, Pg. 16)), $M_t = \lambda_t + (1-\lambda_t)F_t$, where $F_t = \gamma_t\rho_{t-1}F_{t-1} + 1$. Importantly, the $\mathbf{A}_{m_{\text{ETD}}}$ matrix induced by the emphatic weighting is positive semi-definite (Yu, 2015; Sutton et al., 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 et al., 2014) and several others (see Dann et al. (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 $\mathbf{A}_{m,t} \stackrel{\text{def}}{=} \frac{1}{t}\sum_{i=1}^t \mathbf{e}_{m,i}(\mathbf{x}_i - \gamma\mathbf{x}_{i+1})^\top$, and $\mathbf{b}_{m,t} \stackrel{\text{def}}{=} \frac{1}{t}\sum_{i=1}^t \mathbf{e}_{m,i}R_{i+1}$, 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 (Ghavamzadeh et al., 2010).

3. Algorithm derivation

To derive the new algorithm, we first take the gradient of the MSPBE (in 1) to get $\frac{1}{2}\nabla_{\mathbf{w}}\text{MSPBE}(\mathbf{w}, m) = \mathbf{A}_m^\top\mathbf{C}^{-1}(-\mathbf{A}_m\mathbf{w} + \mathbf{b}_m)$. Consider a second order update by computing the Hessian: $\mathbf{A}_m^\top\mathbf{C}^{-1}\mathbf{A}_m^\top$. 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

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t(\mathbf{A}^\top\mathbf{C}^{-1}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{C}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t) = \mathbf{w}_t + \alpha_t\mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$$

In fact, for our quadratic loss, the optimal descent direction is $\mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$ with $\alpha_t = 1$, in the sense that $\text{argmin}_{\Delta\mathbf{w}} \text{loss}(\mathbf{w}_t + \Delta\mathbf{w}) = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$. 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 et al., 2007; Bordes et al., 2009; Mokhtari and Ribeiro, 2014). These methods maintain an approximation to the Hessian, and sample the gradient. The Hessian approximation provides curvature information that can significantly speed convergence, as well as reduce parameter sensitivity to the step-size.

Our objective is to improve on the sample complexity 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 $\mathbf{b} - \mathbf{A}\mathbf{w}$ 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

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (\alpha_t \hat{\mathbf{A}}^\dagger + \eta \mathbf{I})(\mathbf{b} - \mathbf{A}\mathbf{w}_t) \tag{2}$$

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 et al., 2016, Theorem 1). Instead, sampling $\mathbf{b} - \mathbf{A}\mathbf{w}$, 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, providing 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 future research.

We opt for an incremental SVD, that previously proved effective for incremental estimation in reinforcement learning (Gehring et al., 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 step-size. On the other extreme, when $k = d$, ATD is equivalent to an iterative form of LSTD(λ). See the appendix for a further discussion, and implementation details.[1]

4. 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 et al., 2016). Further, ATD should be more effective if $\mathbf{A}$ is low-rank, and so requiring a full-rank $\mathbf{A}$ would limit the practical use-cases for ATD.

1. The appendix can be found at the following url: anonymous link

We first prove convergence of ATD with weightings that give positive semi-definite $\mathbf{A}_m$, and more general settings—both proofs are in the appendix.

Theorem 1. *Under Assumptions 1 and 2 (see appendix), 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* (2) *converges to the fixed-point $\mathbf{w}^\star = \mathbf{A}_m^\dagger \mathbf{b}_m$. Further, if Assumption 3 (see appendix) is satisfied, the convergence rate is*

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

This theorem gives insight into the utility of ATD for speeding convergence. Consider the setting $\alpha = 1$ for ATD, common for second-order methods and let $p = 2$. 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, for a larger k, can pick smaller η and so has a much smaller $\eta^t\lambda_1^{t+1}$, 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.

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 et al., 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 primary role of $\hat{\mathbf{A}}_t$ is to provide curvature information, reducing the algorithms sensitivity to the choice of step-size. The choice of k in ATD is therefore decoupled from the fixed point, and so can be set to balance learning speed and computation with no fear of asymptotic bias.

Because ATD is a stochastic update, not the expected update, we use conventions from stochastic gradient descent to set our parameters, as well as intuition from the theorem. From the convergence rate, for reasonable k, the regularizer η should be small—smaller than a typical stepsize for TD. We set $\alpha_t = \frac{1}{t}$, as in previous stochastic second-order methods (Schraudolph et al., 2007), and set η to a small fixed value, representing a small final step-size and matching the convergence rate intuition.

5. Empirical Results

The results presented in this section were generated from 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.

Our first batch of experiments were conducted on Boyan's chain. 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. Figure 1.1 and 1.2 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 step-sizes and λ. LSTD(λ) using the Sherman-Morrison update (used in many prior empirical studies) is sensitive to the regularization parameter.[2]

Our second batch of experiments were on Mountain car, where perfect approximation of the value function is not possible. 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

2. We are not the first to observe this. Sutton and Barto (2016) note that η plays a role similar to the step-size for LSTD.

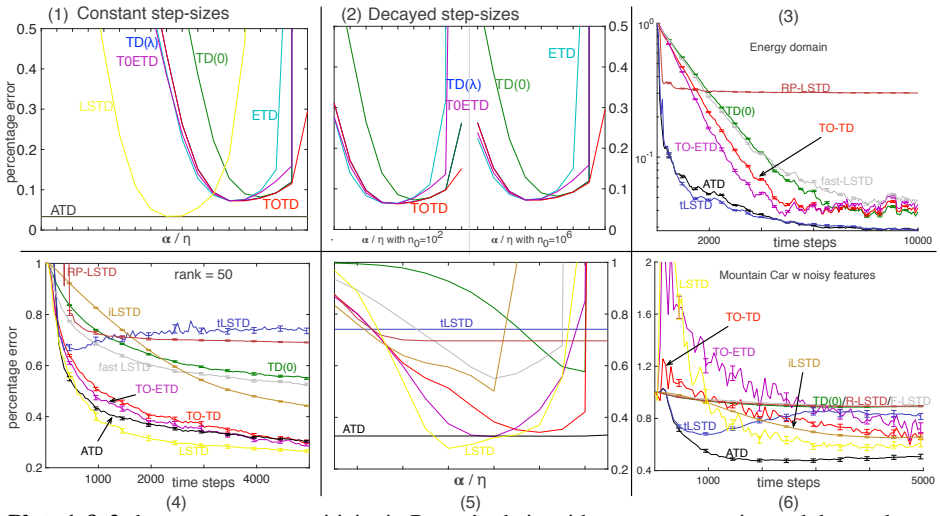

Figure 1: **Plots 1 & 2** show parameter sensitivity in Boyan's chain with constant step-size and decayed step-sizes. Each point summarizes the mean performance (over 1000 time steps) of an algorithm for one setting of α for linear methods, or η for LSTD, and $\alpha/100$ regularizer for ATD, using percentage error compared to the true value function (averaged over 200 runs). In the decayed step-size 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 plot 2. The LSTD algorithm (in yellow plot 1) 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 step-size as $\alpha_t = \frac{1}{t}$ across all experiments). **Plot 4** contains learning curves on the Mountain car domain, showing percentage error versus time steps averaged over 100 runs of ATD with rank 50, LSTD and several baselines described in text. **Plot 5** shows the sensitivity with respect to the learning rate of the linear methods, and regularization parameter of the matrix methods on Mountain Car. The tLSTD algorithm has no parameter besides rank. **Plot 6** shows the learning curve on Mountain Car with noisy features averaged over 100 runs and, **plot 3** shows the energy allocation learning curve, in logscale, averaged over 50 runs

used a fine grain tile coding, resulting in a 1024 dimensional feature representation. The results are summarized in figure 1.4 and 1.5. LSTD and ATD exhibit faster initial learning compared to all other methods, including the sub-quadratic baselines.[3] This is particularly impressive since k is less than 5% of the size of $\mathbf{A}$. The true online linear methods, for example true online TD(λ), 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 1.5); ATD achieved excellent performance with the same parameter setting as we used in Boyan's chain.

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 on each time step. The results are summarized in Figure 1.6 . 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.

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 1.3). The feature vectors were produced via tile coding, resulting in an 8192 dimensional feature vector. 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 both ATD and tLSTD using $r = 40$. The appendix contains additional results for this domain—in the small rank setting, for example, ATD significantly outperforms tLSTD.

3. Fast LSTD Prashanth et al. (2013) uses randomized TD updates in batch; this algorithm can be run incrementally with O(dk) using mini-batches of size k.

References

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

Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 2009.

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

Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: a survey and comparison. *The Journal of Machine Learning Research*, 2014.

Clement Gehring, Yangchen Pan, and Martha White. Incremental Truncated LSTD. In *International Joint Conference on Artificial Intelligence*, 2016.

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

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

H Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta, 2011.

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

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

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

Richard S Sutton, Ashique Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2016.

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

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

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

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

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

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

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

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