Incremental Truncated LSTD

Clement Gehring
MIT CSAIL
Cambridge MA 02139 USA
gehring@csail.mit.edu

Yangchen Pan and **Martha White**
Indiana University
Bloomington IN 47405 USA
{yangpan, martha}@indiana.edu

Abstract

Balancing between computational efficiency and sample efficiency is an important goal in reinforcement learning. Temporal difference (TD) learning algorithms stochastically update the value function, with a linear time complexity in the number of features, whereas least-squares temporal difference (LSTD) algorithms are sample efficient but can be quadratic in the number of features. In this work, we develop an efficient incremental low-rank LSTD(λ) algorithm that progresses towards the goal of better balancing computation and sample efficiency. The algorithm reduces the computation and storage complexity to the number of features times the chosen rank parameter while summarizing past samples efficiently to nearly obtain the sample efficiency of LSTD. We derive a simulation bound on the solution given by truncated low-rank approximation, illustrating a bias-variance trade-off dependent on the choice of rank. We demonstrate that the algorithm effectively balances computational complexity and sample efficiency for policy evaluation in a benchmark task and a high-dimensional energy allocation domain.

1 Introduction

Value function approximation is a central goal in reinforcement learning. A common approach to learn the value function is to minimize the mean-squared projected Bellman error, with dominant approaches generally split into stochastic temporal difference (TD) methods and least squares temporal difference (LSTD) methods. TD learning [Sutton, 1988] requires only $O(d)$ computation and storage per step for d features, but can be sample inefficient [Bradtke and Barto, 1996; Boyan, 1999; Geramifard and Bowling, 2006] because a sample is used only once for a stochastic update. Nonetheless, for practical incremental updating, particularly for high-dimensional features, it remains a dominant approach.

On the other end of the spectrum, LSTD [Bradtke and Barto, 1996] algorithms summarizes all past data into a linear system, and are more sample efficient than TD [Bradtke and Barto, 1996; Boyan, 1999; Geramifard and Bowling, 2006], but at the cost of higher computational complexity and storage complexity. Several algorithms have been proposed to tackle these practical issues,[1] including iLSTD [Geramifard and Bowling, 2006], iLSTD(λ) [Geramifard *et al.*, 2007], sigma-point policy iteration [Bowling and Geramifard, 2008], random projections [Ghavamzadeh *et al.*, 2010], experience replay strategies [Lin, 1993; Prashanth *et al.*, 2013] and forgetful LSTD [van Seijen and Sutton, 2015]. Practical incremental LSTD strategies typically consist of using the system as a model [Geramifard and Bowling, 2006; Geramifard *et al.*, 2007; Bowling and Geramifard, 2008], similar to experience replay, or using random projections to reduce the size of the system [Ghavamzadeh *et al.*, 2010]. To date, however, none seem to take advantage of the fact that the LSTD system is likely low-rank, due to dependent features [Bertsekas, 2007], small numbers of samples [Kolter and Ng, 2009; Ghavamzadeh *et al.*, 2010] or principal subspaces or highways in the environment [Keller *et al.*, 2006].

In this work, we propose t-LSTD, a novel incremental low-rank LSTD(λ), to further bridge the gap between computation and sample efficiency. The key advantage to using a low-rank approximation is to direct approximation to less significant parts of the system. For the original linear system with d features and corresponding $d \times d$ matrix, we incrementally maintain a truncated rank r singular value decomposition (SVD), which reduces storage to significantly smaller $d \times r$ matrices and computation to $O(dr + r^3)$. In addition to these practical computational gains, this approach has several key benefits. First, it exploits the fact that the linear system likely has redundancies, reducing computation and storage without sacrificing much accuracy. Second, the resulting solution is better conditioned, as truncating singular values is a form of regularization. Regularization strategies have proven effective for stability [Bertsekas, 2007; Farahmand *et al.*, 2008; Kolter and Ng, 2009; Farahmand, 2011]; however, unlike these previous approaches, the truncated SVD also reduces the size of the system. Third, like iLSTD, it provides a close approximation to the system, but with storage complexity reduced to O(dr) instead of O(d^2) and a more intuitive toggle r to balance computation and approximation. Finally, the ap-

[1] A somewhat orthogonal strategy is to sub-select features before applying LSTD [Keller *et al.*, 2006]. Feature selection is an important topic on its own; we therefore focus exploration on direct approximations of the LSTD system itself.

proach is more promising for tracking and for control, because previous samples can be efficiently down-weighted in O(r) and the solution can be computed in O(dr) time, enabling every-step updating.

To better investigate the merit of low-rank approximations for LSTD, we first derive a new simulation bound for low-rank approximations, highlighting the bias-variance trade-off given by this form of regularization. We then empirically investigate the rank properties of the system in a benchmark task (Mountain Car) with common feature representations (tile coding and RBFs), to explore the validity of using low-rank approximation in reinforcement learning. Finally, we demonstrate efficacy of t-LSTD for value function approximation in this domain as well as a high-dimensional energy allocation domain.

2 Problem formulation

We assume the agent interacts with and receives reward from an environment formalized by a Markov decision process: $(\mathcal{S}, \mathcal{A}, \Pr, r, \gamma)$ where $\mathcal{S}$ is the set of states, $n = |\mathcal{S}|$; $\mathcal{A}$ is the set of actions; $\Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, where $\Pr(s, a, s')$ is the probability of transitioning from state s into state s' when taking action a, receiving reward $r(s, a, s')$; and $\gamma \in [0, 1]$ is the discount rate. For a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $\sum_{a \in \mathcal{A}} \pi(s, a) = 1 \; \forall s \in \mathcal{S}$, define matrix $\mathbf{P}^\pi \in \mathbb{R}^{n \times n}$ as $\mathbf{P}^\pi(s, s') = \sum_{a \in \mathcal{A}} \pi(s, a) \Pr(s, a, s')$ and vector $\mathbf{r}_\pi \in \mathbb{R}^n$ as the average rewards from each state under π. The value at a state s_t is the expected discounted sum of future rewards, assuming actions are selected according to π,

$$V^\pi(s_t) = \mathbf{r}_\pi(s_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathbf{P}^\pi(s_t, s_{t+1}) V^\pi(s_{t+1}).$$

Value function learning using linear function approximation can be expressed as a linear system [Bradtke and Barto, 1996]: $\mathbf{A}\mathbf{w} = \mathbf{b}$ for

$$\mathbf{A} = \mathbf{X}^\top \mathbf{D} (\mathbf{I} - \gamma\lambda \mathbf{P}^\pi)^{-1} (\mathbf{I} - \gamma \mathbf{P}^\pi) \mathbf{X}$$
$$\mathbf{b} = \mathbf{X}^\top \mathbf{D} (\mathbf{I} - \gamma\lambda \mathbf{P}^\pi)^{-1} \mathbf{r}_\pi$$

where each row in $\mathbf{X} \in \mathbb{R}^{n \times d}$ corresponds to the features for a state; $\mathbf{D}$ is a diagonal matrix with the stationary distribution of π on the diagonal; and λ is the trace parameter for the λ-return. For action-value function approximation, the system is the same, but with state-action features in $\mathbf{X}$. These matrices are approximated using

$$\mathbf{A}_T = \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \quad \text{and} \quad \mathbf{b}_T = \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t r_{t+1}$$

for eligibility trace $\mathbf{z}_t = \sum_{i=0}^{t} (\gamma\lambda)^{t-i} \mathbf{x}_i$ and sampled trajectory $s_0, a_0, r_1, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$.

There are several strategies to solve this system incrementally. A standard approach is to use TD and variants, which stochastically update $\mathbf{w}$ with new samples as $\mathbf{w} = \mathbf{w} + \alpha(r_{t+1} + \gamma \mathbf{x}_{t+1}^\top \mathbf{w} - \mathbf{x}_t^\top \mathbf{w}) \mathbf{z}_t$. The LSTD algorithms instead incrementally approximate these matrices or corresponding system. For example, the original LSTD algorithm [Bradtke and Barto, 1996] incrementally maintains $\mathbf{A}_t^{-1}$ using the matrix inversion lemma so that on each step the new solution $\mathbf{w} = \mathbf{A}_t^{-1} \mathbf{b}_t$ can be computed.

We iteratively update and solve this system by maintaining a low rank approximation to $\mathbf{A}_t$ directly. Any matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ has a singular value decomposition (SVD) $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$, where $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is a diagonal matrix of the singular values of $\mathbf{A}$ and $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times d}$ are orthonormal matrices: $\mathbf{U}^\top \mathbf{U} = \mathbf{I} = \mathbf{V}^\top \mathbf{V}$ and $\mathbf{U}\mathbf{U}^\top = \mathbf{I} = \mathbf{V}\mathbf{V}^\top$. With this decomposition, for full rank $\mathbf{A}$, the inverse of $\mathbf{A}$ is simply computed by inverting the singular values, to get $\mathbf{w} = \mathbf{A}^{-1} \mathbf{b} = \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{U}^\top \mathbf{b}$. In many cases, however, the rank of $\mathbf{A}$ may be smaller than d, giving $d - \text{rank}(\mathbf{A})$ singular values that are zero. Further, we can approximate $\mathbf{A}$ by dropping (i.e., zeroing) some number of the smallest singular values, to obtain a rank r approximation. Correspondingly, rows of $\mathbf{U}$ and $\mathbf{V}$ are zeroed, reducing the size of these matrices to $d \times r$. The further we reduce the dimension, the more practical for efficient incremental updating; however there is clearly a trade-off in terms of accuracy of the solution. We first investigate the theoretical properties of using a low-rank approximation to $\mathbf{A}_t$ and then present the incremental t-LSTD algorithm.

3 Characterizing the low-rank approximation

Low-rank approximations provide an efficient approach to obtaining stable solutions for linear systems. The approach is particularly well motivated for our resource constrained setting, because of the classical Eckart-Young-Mirsky theorem [Eckart and Young, 1936; Mirsky, 1960], which states that the optimal rank r approximation to a matrix under any unitarily invariant norm (e.g., Frobenius norm, spectral norm, nuclear norm) is the truncated singular value decomposition. In addition to this nice property, which facilitates development of an efficient approximate LSTD algorithm, the truncated SVD can be viewed as a form of regularization [Hansen, 1986], improving the stability of the solution.

To see why truncated SVD regularizes the solution, consider the solution to the linear system

$$\mathbf{w} = \mathbf{A}^\dagger \mathbf{b} = \mathbf{V} \mathbf{\Sigma}^\dagger \mathbf{U}^\top \mathbf{b} = \sum_{i=1}^{\text{rank}(\mathbf{A})} \frac{\mathbf{v}_i \mathbf{u}_i^\top}{\sigma_i} \mathbf{b}$$

for ordered singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_{\text{rank}(\mathbf{A})} > \sigma_{\text{rank}(\mathbf{A})+1} = 0, \ldots, \sigma_d = 0$. $\mathbf{A}^\dagger$ is the pseudo-inverse of $\mathbf{A}$, with $\mathbf{\Sigma}^\dagger = \text{diag}(\sigma_1^{-1}, ..., \sigma_{\text{rank}(\mathbf{A})}^{-1} 0, ..., 0)$ composed of the inverses of the non-zero singular values. For very small, but still non-zero σ_i, the outer product $\mathbf{v}_i \mathbf{u}_i^\top$ will be scaled by a large number; this will often correspond to highly overfitting the observed samples and a high variance estimate. A common practice is to regularize $\mathbf{w}$ with $\eta \|\mathbf{w}\|_2$ for regularization weight $\eta > 0$, modifying the multiplier from σ_i^{-1} to $\sigma_i / (\sigma_i^2 + \eta)$ because $\mathbf{w} = (\mathbf{A}^\top \mathbf{A} + \eta \mathbf{I})^{-1} \mathbf{A}^\top \mathbf{b} = \mathbf{V} (\mathbf{\Sigma}^2 + \eta \mathbf{I})^{-1} \mathbf{\Sigma} \mathbf{U}^\top \mathbf{b}$. The regularization reduces variance but introduces bias controlled by η; for $\eta = 0$, we obtain the unbiased solution. Similarly, by thresholding the smallest singular values to retain only the top r singular values,

$$\mathbf{w} = \mathbf{A}_r^\dagger \mathbf{b} = \mathbf{V} \text{diag}(\sigma_1^{-1}, ..., \sigma_r^{-1}, 0, ..., 0) \mathbf{U}^\top \mathbf{b} = \sum_{i=1}^{r} \frac{\mathbf{v}_i \mathbf{u}_i^\top}{\sigma_i} \mathbf{b}$$

we introduce bias, but reduce variance because the size of σ_r^{-1} can be controlled by the choice of $r < \text{rank}(\mathbf{A})$.

To characterize the bias-variance tradeoff, we bound the difference between the true solution, $\mathbf{w}^*$, and the approximate rank r solution at time t, $\mathbf{w}_{t,r}$. We use a similar analysis to the one used for regularized LSTD [Bertsekas, 2007, Proposition 6.3.4]. This previous bound does not easily extend, because in regularized LSTD, the singular values are scaled up, maintaining the information in the singular vectors (i.e., no columns are dropped from $\mathbf{U}$ or $\mathbf{V}$). We bound the loss incurred by dropping singular vectors using insights from work on ill-posed systems.

The following is a simple but realistic assumption for ill-posed systems [Hansen, 1990]. The assumption states that $\mathbf{u}_i^\top \mathbf{b}$ shrinks faster than σ_i^p, where p specifies the smoothness of the solution $\mathbf{w}$ and is related to the smoothness parameter for the Hilbert space setting [Groetsch, 1984, Cor. 1.2.7].

Assumption 1: The linear system defined by $\mathbf{A} = \mathbf{U\Sigma V}^\top$ and $\mathbf{b}$ satisfy the *discrete Picard condition*: for some $p > 1$,

$$\begin{aligned} |\mathbf{u}_i^\top \mathbf{b}| &\le \sigma_i^p && \text{for } i = 1, \ldots, \operatorname{rank}(\mathbf{A}) \\ |\mathbf{u}_i^\top \mathbf{b}| &\le \sigma_{\operatorname{rank}(A)}^p && \text{for } i = \operatorname{rank}(\mathbf{A}) + 1, \ldots, d. \end{aligned}$$

Assumption 2: As $t \to \infty$, the sample average $\mathbf{A}_t$ converges to the true $\mathbf{A}$. This assumption can be satisfied with typical technical assumptions (see [Tsitsiklis and Van Roy, 1997]).

We write the SVD of $\mathbf{A} = \mathbf{U\Sigma V}^\top$ and $\mathbf{A}_t = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\top$, where to avoid cluttered notation, we do not explicitly subscript with t. Further, though the singular values are unique, there is a space of equivalent singular vectors, up to sign changes and multiplication by rotation matrices. We assume that among the space of equivalent SVDs of $\mathbf{A}_t$, the most similar singular vectors for each singular value are chosen between $\mathbf{A}$ and $\mathbf{A}_t$. This avoids uniqueness issues without losing generality, because we only conceptually compare the SVDs of $\mathbf{A}$ and $\mathbf{A}_t$; the proof does not rely on practically obtaining this matching SVD.

Theorem 1 (Bias-variance trade-off of rank-r approximation)**.** *Let $\mathbf{A}_{t,r} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}_r\hat{\mathbf{V}}^\top$ be the approximated $\mathbf{A}$ after t samples, truncated to rank r, i.e., with the last $r+1, \ldots, d$ singular values zeroed. Let $\mathbf{w}^* = \mathbf{A}^\dagger \mathbf{b}$ and $\mathbf{w}_{t,r} = \mathbf{A}_{t,r}^\dagger \mathbf{b}_t$. Under Assumption 1 and 2, the relative error of the rank-r weights to the true weights $\mathbf{w}^*$ is bounded as follows:*

$$\begin{aligned} \|\mathbf{w}_{t,r} - \mathbf{w}^*\|_2 \le \frac{1}{\hat{\sigma}_r}\|\mathbf{b}_t - \mathbf{A}_t\mathbf{w}^*\|_2 + (d-r)\epsilon(t) \\ + \underbrace{(d-r)\sigma_r^{p-1}}_{\textit{bias}} \end{aligned}$$

for function $\epsilon : \mathbb{N} \to [0, \infty)$, where $\epsilon(t) \to 0$ as $t \to \infty$:

$$\begin{aligned} \epsilon(t) = \min\Big(&\textit{rank}(\mathbf{A})\sigma_1^{p-1}, \\ &\sum_{j=1}^{\textit{rank}(\mathbf{A})} \left\|\mathbf{v}_j\sigma_j^{p-1} - \hat{\mathbf{v}}_j\hat{\sigma}_j^{p-1}\right\|_2 + \hat{\sigma}_r^{p-1} - \sigma_r^{p-1}\Big). \end{aligned}$$

A detailed proof is provided in an appendix[Gehring and White, 2015]. The key step is to split up the error into two terms: approximation error due to a finite number of samples t and bias due the choice of $r < d$. Then the second part is bounded using the discrete Picard condition to ensure that the magnitude of $\mathbf{u}_j^\top \mathbf{b}$ does not dominate the error, and by adding and subtracting terms to express the error in terms of differences between $\mathbf{A}$ and $\mathbf{A}_t$. Because $\mathbf{A}_t$ converges to $\mathbf{A}$, we can see that $\epsilon(t)$ converges to zero because the differences $\mathbf{v}_j\sigma_j^{p-1} - \hat{\mathbf{v}}_j\hat{\sigma}_j^{p-1}$ and $\hat{\sigma}_r^{p-1} - \sigma_r^{p-1}$ converge to zero.

Remark 1: Notice that for no truncation, the bias term disappears and the first term could be very large because $\hat{\sigma}_r = \hat{\sigma}_d$ could be very small (and often is for systems studied to-date, including in the below experiments). In fact, previous work on finite sample analysis of LSTD uses an unbiased estimate and the bound suffers from an inverse relationship to the smallest eigenvalue of $\mathbf{X}^\top\mathbf{X}$ (see [Lazaric *et al.*, 2010, Lemma 3], [Ghavamzadeh *et al.*, 2010; Tagorti and Scherrer, 2015]). Here, we avoid such a potentially large constant in the bound at the expense of an additional bias term determined by the choice of r. Lasso-TD [Ghavamzadeh and Lazaric, 2011] similarly avoids such a dependence, using ℓ_1 regularization; to the best of our knowledge, however, there does not yet exist an efficient incremental Lasso-TD algorithm. A future goal is to use the above bound, to obtain a finite sample bound for t-LSTD(λ), using the most up-to-date analysis by Tagorti and Scherrer [2015] and more general techniques for linear system introduced by Pires and Szepesvari [2012].

Remark 2: The discrete Picard condition could be relaxed to an average discrete Picard condition, where $|\mathbf{u}_i^\top \mathbf{b}|$ on average is similar to σ_i^{-1}, with a bound on the variance of this ratio. The assumption above, however, simplifies the analysis and much more clearly illustrates the importance of the decay of $\mathbf{u}_i^\top \mathbf{b}$ for obtaining stable LSTD solutions.

4 Incremental low-rank LSTD(λ) algorithm

We have shown that a low-rank approximation to $\mathbf{A}_t$ is effective for computing the solution to LSTD from t samples. However, the computational complexity of explicitly computing $\mathbf{A}_t$ from samples and then performing a SVD is $O(d^3)$, which is not feasible for most settings. In this section, we propose an algorithm that incrementally computes a low-rank singular value decomposition of $\mathbf{A}_t$, from samples, with significantly improved storage O(dr) and computational complexity $O(dr + r^3)$, which we can further reduced to O(dr) using mini-batches of size r.

To maintain a low-rank approximation to $\mathbf{A}_t$ incrementally, we need to update the SVD with new samples. With each new $\mathbf{x}_t$, we add the rank-one matrix $\mathbf{z}_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})^\top$ to $\mathbf{A}_t$. Consequently, we can take advantage of recent advances for fast low-rank SVD updates [Brand, 2006], with some specialized computational improvements for our setting. Algorithm 1 summarizes the generic incremental update for t-LSTD, which can use mini-batches or update on each step, depending on the choice of the mini-batch size k. Due to space constraints, the detailed pseudo-code for the SVD updates are left out but detailed code and explanations will be published on-line. The basics of the SVD update follow from previous work [Brand, 2006] but our implementation offers some optimizations specific for the LSTD case.

By maintaining the SVD incrementally, we do not need to explicitly maintain $\mathbf{A}_t$; therefore, storage is reduced to the size of the truncated singular vector matrices, which is

Algorithm 1 t-LSTD(λ) using incremental SVD

```
// Input rank $r$, and mini-batch size $k$
// with differing update-svd for $k = 1$ and $k > 1$
$\mathbf{U} \leftarrow [], \mathbf{V} \leftarrow [], \mathbf{\Sigma} \leftarrow 0, \mathbf{b} \leftarrow \mathbf{0}, \mathbf{z} \leftarrow \mathbf{0}, i \leftarrow 0, t \leftarrow 1$
$\mathbf{x} \leftarrow$ the initial observation
repeat
    Take action according to $\pi$, observe $\mathbf{x}'$, reward $r$
    $\beta \leftarrow 1/(t+k)$
    $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \mathbf{x}$
    $\mathbf{d} \leftarrow \beta(\mathbf{x} - \gamma\mathbf{x}')$
    $\mathbf{Z}_{:,i} \leftarrow \mathbf{z}$
    $\mathbf{D}_{:,i} \leftarrow \mathbf{d}$
    $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta\mathbf{z}r$
    $i \leftarrow i + 1$
    if $i \geq k$ then
        // Returns $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times r}$, diagonal $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$
        $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V} \leftarrow$
        update-svd($\mathbf{U}, (1-\beta)\mathbf{\Sigma}, \mathbf{V}, \sqrt{\beta}\mathbf{Z}, \sqrt{\beta}\mathbf{D}, r$)
        $\mathbf{Z} \leftarrow 0^{d \times k}, \mathbf{D} \leftarrow 0^{d \times k}, i \leftarrow 0, t \leftarrow t + k$
    end if
    $\mathbf{w} \leftarrow \mathbf{V}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\top}\mathbf{b}$ // $O(dr)$ time
until agent done interaction with environment
```

O(dr). To maintain $O(dr)$ computational complexity, matrix and vector multiplications need to be carefully ordered. For example, to compute $\mathbf{w}$, first $\tilde{\mathbf{b}} = \mathbf{U}^{\top}\mathbf{b}$ is computed in $O(dr)$, then $\mathbf{\Sigma}_r\tilde{\mathbf{b}}$ is computed in $O(r)$, and finally that is multiplied by $\mathbf{V}$ in $O(dr)$. For $k = 1$ (update on each step), the $O(r^3)$ computation arises from a re-diagonalization and the multiplication of the resulting orthonormal matrices. For mini-batches of size $k = r$, we can get further computational improvements by amortizing costs across r steps, to obtain a total amortized complexity $O(dr)$, losing the r^3 term.

As an additional benefit, unlike previous incremental LSTD algorithms, we maintain normalized $\mathbf{A}_t$ and $\mathbf{b}_t$, by incorporating the term β. On each step, we use

$$\mathbf{A}_{t+1} = \tfrac{1}{t+1}(t\mathbf{A}_t + \mathbf{z}_t\mathbf{d}_t^{\top}) = (1-\beta_t)\mathbf{A}_t + \beta_t\mathbf{z}_t\mathbf{d}_t^{\top}$$

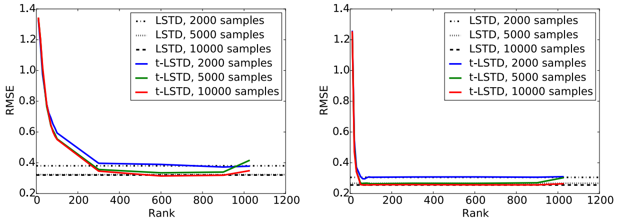
for $\beta_t = \frac{1}{t+1}$. The multiplication of $\mathbf{A}_t$ by $1 - \beta_t$ requires only O(r) computation because $(1-\beta_t)\mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^{\top} = \mathbf{U}_r(1-\beta_t)\mathbf{\Sigma}_r\mathbf{V}_r^{\top}$. Multiplying the full $\mathbf{A}$ matrix by $1 - \beta_t$, on the other hand, would require O(d^2) computation, which is prohibitive. Further, β_t can be selected to obtain a running average, as in Algorithm 1, or more generally can be set to any $\beta_t \in (0, 1)$. For example, to improve tracking, β can be chosen as a constant to weight more recent samples more highly in the value function estimate.

5 Experiments

We empirically investigate t-LSTD, for $k = r$ in a benchmark domain and $k = 1$ in an energy allocation domain.

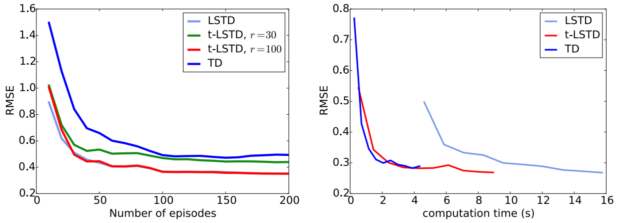
Value function accuracy in benchmark domains:

We first investigate the performance of t-LSTD in the Mountain Car benchmark. The goal in this setting is to carefully investigate t-LSTD against the two extremes of TD and LSTD, and evaluate the utility for balancing sample and computational complexity. We use two common feature representations: tile coding and radial basis function (RBF) coding. We set the policy to the commonly used energy-pumping policy, which picks actions by pushing along the current velocity. The true values are estimated by using rollouts from states chosen in a uniform 20x20 grid of the state-space. The reported root mean squared error (RMSE) is computed between the estimated value functions and the rollout values. The tile coding representation has 1000 features, using 10 layers of 10x10 grids. The RBF representation has 1024 features, for a grid of 32x32 RBFs with width equal to 0.12 times the total range of the state space. We purposefully set the total number of features to be similar in both cases in order to keep the results comparable. We set the RBF widths to obtain



(a) Rank versus performance with tile coding (b) Rank versus performance with RBFs

Figure 1: The impact of the rank r on RMSE of the true discounted returns and the learned value function in Mountain Car. We can see that large r are not necessary, with performance levelling off at $r = 50$. For high values of r and fewer samples, the error slightly increases, likely due to some instability with incremental updating and very small singular values.



(a) RMSE versus samples (b) RMSE versus runtime

Figure 2: RMSE of the true discounted returns and the learned value function in Mountain Car with RBFs. For (a) we can see that with a significantly reduced r, t-LSTD can match LSTD, and outperforms TD. This is the best setting for LSTD, where computation is not restricted, and it can spend time processing samples. For (b) we provide the best scenario for TD, with unlimited samples. Once again, t-LSTD can almost match the performance of TD, and significantly outperforms LSTD. Together, these graphs indicate that t-LSTD can balance between the two extremes. The reported results are for the best parameter settings for TD, and for $r = 100$ and $\lambda = 0$ for t-LSTD.

good performance from LSTD. The other parameters (λ and step-size) are optimized for each algorithm. In the Mountain Car results, we use the mini-batch case where $k = r$ and a discount $\gamma = 0.99$. Results are averaged over 30 runs.

Empirically, we observed that the $\mathbf{A}$ has only a few large singular value with the rest being small. This was observed in Mountain Car across a wide range of parameter choices for both tile coding and RBFs, hinting that $\mathbf{A}$ could be reasonably approximated with small rank. In order to investigate the effect of the rank of t-LSTD , we vary r and run t-LSTD on some fix number of samples. In Figure 1 (a) and (b), we observe a gracious decay in the quality of the estimated value function as the rank is reduced while achieving LSTD level performance with as little as $r = 50$ for RBFs ($d = 1024$) and $r = 300$ for tile coding ($d = 1000$).

Given large enough rank and numerical precision, LSTD and t-LSTD should behave similarly. To verify this, in Figure 2 (a), we plot the learning curves of t-LSTD in the case where the r is too small and the case where r is large enough, alongside LSTD and TD. As expected, we observe LSTD and t-LSTD to have near identical learning curves for $r = 100$, while, for the case with smaller rank $r = 30$, we see the algorithm converge rapidly to an inferior solution. TD is less sample efficient and so converges more slowly than either.

Sample efficiency is an important property for an algorithm but does not completely capture the needs of an engineer attempting to solve a domain. In many case, the requirements tend to call for a balance between runtime and number of samples. In cases where a simulator is available, such as in game playing (e.g., atari, chess, backgammon, go), samples are readily available and only computational cost matters. For this reason, we explore the performance of TD, LSTD, and t-LSTD when given unlimited data but limited CPU time. In Figure 2 (b), we plot the accuracy of the methods with respect to computation time used. The algorithms are given access to varying amounts of samples: up to 8000 samples for TD and up to 4000 for t-LSTD and LSTD. The RMSE and time taken is monitored, after which, the points are averaged to generate the plots comparing runtime to the error in the learned solution.

These results show that TD, despite poor sample efficiency, outperforms LSTD for a given runtime, due to the computational efficiency of each update. This supports the trend of preferring TD for large problems over LSTD. We observe t-LSTD achieve a comparable runtime to TD. Even though t-LSTD is computationally more costly than TD, its superior sample efficiency compensates. Furthermore, this infinite sample stream case is favorable to TD. In a scenario where data is obtain in real-time, sacrificing sample efficiency for computational gains might leave TD idling occasionally, further reinforcing t-LSTD as a good alternative.

These results indicate that t-LSTD offers an effective approach to balance sample efficiency and computational efficiency to match both TD and LSTD in their respective use cases, offering good performance when data is plentiful while still offering LSTD-like sample efficiency.

Value function accuracy in an energy domain

In this section, we demonstrate the performance of the fully incremental algorithm ($k = 1$) in a large energy allocation domain [Salas and Powell, 2013]. The focus in this experiment is to evaluate the practical utility of t-LSTD in an important application, versus realistic competitors: TD[2] and iLSTD. The goal of the agent in this domain is to maximize revenue and satisfy demand. Each action vector is an allocation decision. Each state is a four dimensional variable: the amount of energy in storage, the amounts of renewable generation available, the market price of energy, and the demand needs to be satisfied. We use a provided near-optimal policy [Salas and Powell, 2013]. We set $\gamma = 0.8$.

To approximate the value function, we use tile coding with 32 tilings where each tiling contains $5 \times 5 \times 10 \times 5$ grids, resulting in $40,000$ features and also included a bias unit. We choose this representation, because iLSTD is only computationally feasible for high-dimensional *sparse* representations. As before, extensive rollouts are computed from a subset of states, to compute accurate estimate of the true value, and then stored for comparison in the computation of the RMSE. Results were averaged over 30 runs.

We report results for several values of r for t-LSTD. We sweep the additional parameters in the other algorithms, including step-sizes α for TD and iLSTD and m for iLSTD. We sweep a range of $\alpha_0 = \{2^{-11}, 2^{-10}, 2^{-9}, ..., 2^{-1}\}$, and divide by the number of active features (which in this case is 2^6). Further, because iLSTD is unstable unless α is decayed,we further sweep the decay formula as suggested by Geramifard and Bowling [2006]

$$\alpha_t = \alpha_0 \frac{N_0 + 1}{N_0 + t},$$

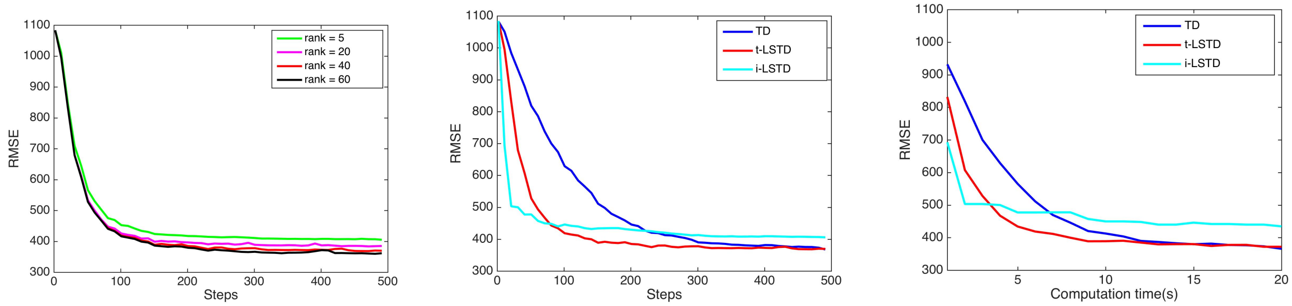
where N_0 is chosen from $\{10, 100, 1000\}$. To focus parameter sweeps on the step-size, which had much more effect for iLSTD, we set $\lambda = 0.9$ for all other algorithms, except for tLSTD which we set $\lambda = 1.0$. We choose $r \in \{5, 20, 40, 60\}$ and $m \in \{10, 20, 30, 40, 50\}$. We restrict the iLSTD parameters to a small set, since there are too many options, even the optimal stepsize would be different when we choose different m values. Preliminary investigation indicated that 50 was large enough for iLSTD.

In this domain, under the common strategy of creating a large number of fixed features (tile coding or RBFs), t-LSTD is able to significantly take advantage of low rank structure, learning more efficiently without incurring much computational cost. Figure 3 shows that t-LSTD performs well with a small $r = 40 << d = 40,000$, and outperforms both TD and iLSTD.

We highlight that iLSTD is one of the only practical competitors introduced for this setting: incremental learning with computational constraints. Even then, iLSTD is restrictive in that the feature representation must be sparse and its storage requirements are O(d^2). Further, though it was reasonably robust to the choice of m, we found iLSTD was quite sensitive to the choice of step-size parameter. In fact, without a careful decay, we still encountered divergence issues.

The goal here was to investigate the performance of the simplest version of t-LSTD, with fewest parameters and with-

[2] We also compared to true-online TD [van Seijen and Sutton, 2014], but it gave very similar performance; we therefore omit it.



(a) RMSE vs samples, rank comparison (b) RMSE vs samples, algorithm comparison (c) RMSE vs runtime

Figure 3: RMSE of the value function in the energy allocation domain. **(a)** Performance of t-LSTD improves as the rank increases; however, even for small $r = 5$, the algorithm still converges with some bias. For r smaller than 5, the error was significantly worse. **(b)** With $r = 40$, t-LSTD converges to the almost same level with TD in significantly fewer steps. The best parameters are chosen for each algorithm, m = 50 for iLSTD and r = 40 for tLSTD. **(c)** As before, we plot RMSE versus runtime, but now by selecting a scenario in between the extremes plotted in Figure 2. The number of samples per second is restricted to 25 samples, meaning TD is sometimes idle waiting for more samples, and iLSTD (m = 50) and t-LSTD (r = 40) could be too slow to process all the samples. This plot further indicates the advantages of t-LSTD, particularly as it is faster than TD in terms of sample efficiency and scales better than iLSTD and converges to a better solution.

out optimizing thresholds, which were kept fixed at reasonable heuristics across all experiments. This choice does impact the learning curves of t-LSTD. For example, though t-LSTD has significantly faster early convergence, it is less smooth than either TD or iLSTD. This lack of smoothness could be due to not optimizing these parameters and further because $\mathbf{w}_t$ is solved on each step. Beyond this vanilla implementation of t-LSTD, there are clear avenues to explore to more smoothly update $\mathbf{w}_t$ with the low-rank approximation to $\mathbf{A}$. Nonetheless, even in its simplest form, t-LSTD provides an attractive alternative to TD, obtaining sample efficiency improvements without much additional computation and without the need to tune a step-size parameter.

6 Discussion and conclusion

This paper introduced an efficient value function approximation algorithm, called t-LSTD(λ), that maintains an incremental truncated singular value decomposition of the LSTD matrix. We systematically explored the validity of using low-rank approximations for LSTD, first by proving a simulation error bound for truncated low-rank LSTD solutions and then, empirically, by examining an incremental truncated LSTD algorithm in two domains. We demonstrated performance of t-LSTD in the benchmark domain, Mountain Car, exploring runtime properties and the effect of the small rank approximation, and in a high-dimensional energy allocation domain, illustrating that t-LSTD enables a nice interpolation between the properties of TD and LSTD, and out-performs iLSTD.

There are several potential benefits of t-LSTD that we did not yet explore in this preliminary investigation. First, there are clear advantages to t-LSTD for tracking and control. As mentioned above, unlike previous LSTD algorithms, past samples for t-LSTD can be efficiently down-weighted with a $\beta_t \in (0, 1)$. By enabling down-weighting, $\mathbf{A}_t$ is more strongly influenced by recent samples and so can better adapt in a non-stationary environment, such as for control.

Another interesting avenue is to take advantage of t-LSTD for early learning, to improve sample efficiency, and then switch to TD to converge to an unbiased solution. Even for highly constrained systems in terms of storage and computation, aggressively small r can still be useful for early learning. Further empirical investigation could give insight into when this switch could occur, depending on the choice of r.

Finally, an important avenue for this new approach is to investigate the convergence properties of truncated incremental SVDs. The algorithm derivation requires only simple algebra and is clearly sound; however, to the best of our knowledge, the question of convergence under numerical stability and truncating non-zero singular values remains open. The truncated incremental SVD has been shown to be practically useful in numerous occasions, such as for principal components analysis and partial least squares [Arora *et al.*, 2012]. Moreover, there are some informal arguments (using randomized matrix theory) that even under truncation the SVD will re-orient [Brand, 2006]. This open question is an important next step in understanding t-LSTD and, more generally, incremental singular value decomposition algorithms for reinforcement learning.

References

[Arora *et al.*, 2012] R Arora, A Cotter, K Livescu, and N Srebro. Stochastic optimization for PCA and PLS. In *Annual Allerton Conference on Communication, Control, and Computing*, 2012.

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

[Bowling and Geramifard, 2008] M Bowling and A Geramifard. Sigma point policy iteration. In *International Conf. on Autonomous Agents and Multiagent Systems*, 2008.

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

[Bradtke and Barto, 1996] Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 1996.

[Brand, 2006] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 2006.

[Eckart and Young, 1936] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936.

[Farahmand *et al.*, 2008] A M Farahmand, M Ghavamzadeh, and C Szepesvári. Regularized policy iteration. In *Advances in Neural Information Processing Systems*, 2008.

[Farahmand, 2011] A Farahmand. *Regularization in reinforcement learning*. PhD thesis, Univ. of Alberta, 2011.

[Gehring and White, 2015] Clement Gehring and Martha White. Incremental truncated LSTD. *CoRR*, abs/1511.08495, 2015.

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

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

[Ghavamzadeh and Lazaric, 2011] M Ghavamzadeh and A Lazaric. Finite-sample analysis of Lasso-TD. In *International Conference on Machine Learning*, 2011.

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

[Groetsch, 1984] C W Groetsch. *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*. Pitman Advanced Publishing Program, 1984.

[Hansen, 1986] P C Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 1986.

[Hansen, 1990] Per Christian Hansen. The discrete picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*, 1990.

[Keller *et al.*, 2006] PW Keller, S Mannor, and D Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *International Conference on Machine Learning*, 2006.

[Kolter and Ng, 2009] JZ Kolter and AY Ng. Regularization and feature selection in least-squares temporal difference learning. In *Inter. Conf. on Machine Learning*, 2009.

[Lazaric *et al.*, 2010] A Lazaric, M Ghavamzadeh, and R Munos. Finite sample analysis of LSTD. *International Conference on Machine Learning*, 2010.

[Lin, 1993] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, CMU, 1993.

[Mirsky, 1960] L Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quartely Journal Of Mathetmatics*, 1960.

[Pires and Szepesvari, 2012] Bernardo Avila Pires and Csaba Szepesvari. Statistical linear estimation with penalized estimators: an application to reinforcement learning. In *International Conference on Machine Learning*, 2012.

[Prashanth *et al.*, 2013] L A Prashanth, Nathaniel Korda, and Rémi Munos. Fast LSTD using stochastic approximation: Finite time analysis and application to traffic control. *arXiv.org*, 2013.

[Salas and Powell, 2013] D F Salas and W B Powell. Benchmarking a Scalable Approximate Dynamic Programming Algorithm for Stochastic Control of Multidimensional Energy Storage Problems. *Dept Oper Res Financial Eng*, 2013.

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

[Tagorti and Scherrer, 2015] Manel Tagorti and Bruno Scherrer. On the Rate of Convergence and Error Bounds for LSTD(λ). In *Inter. Conf. on Machine Learning*, 2015.

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

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

[van Seijen and Sutton, 2015] H van Seijen and R.S. Sutton. A deeper look at planning as learning from replay. In *International Conference on Machine Learning*, 2015.