Effective sketching methods for value function approximation

Yangchen Pan, Erfan Sadeqi Azer and Martha White
Department of Computer Science
Indiana University Bloomington
yangpan@iu.edu, esadeqia@iu.edu, martha@indiana.edu

Abstract

High-dimensional representations, such as radial basis function networks or tile coding, are common choices for policy evaluation in reinforcement learning. Learning with such high-dimensional representations, however, can be expensive, particularly for matrix methods, such as least-squares temporal difference learning or quasi-Newton methods that approximate matrix step-sizes. In this work, we explore the utility of sketching for these two classes of algorithms. We highlight issues with sketching the high-dimensional features directly, which can incur significant bias. As a remedy, we demonstrate how to use sketching more sparingly, with only a left-sided sketch, that can still enable significant computational gains and the use of these matrix-based learning algorithms that are less sensitive to parameters. We empirically investigate these algorithms, in four domains with a variety of representations. Our aim is to provide insights into effective use of sketching in practice.

1 INTRODUCTION

A common strategy for function approximation in reinforcement learning is to overparametrize: generate a large number of features to provide a sufficiently complex function space. For example, one typical representation is a radial basis function network, where the centers for each radial basis function are chosen to exhaustively cover the observation space. Because the environment is unknown—particularly for the incremental learning setting—such an overparameterized representation is more robust to this uncertainty because a reasonable representation is guaranteed for any part of the space that might be visited. Once interacting with the environment, however, it is likely not all features will become active, and that a lower-dimensional subspace will be visited.

A complementary approach for this high-dimensional representation expansion in reinforcement learning, therefore, is to use projections. In this way, we can overparameterize for robustness, but then use a projection to a lower-dimensional space to make learning feasible. For an effectively chosen projection, we can avoid discarding important information, and benefit from the fact that the agent only visits a lower-dimensional subspace of the environment in the feature space.

Towards this aim, we investigate the utility of sketching: projecting with a random matrix. Sketching has been extensively used for efficient communication and solving large linear systems, with a solid theoretical foundation and a variety of different sketches (Woodruff, 2014). Sketching has been previously used in reinforcement learning, specifically to reduce the dimension of the features. Bellemare et al. (2012) replaced the standard biased hashing function used for tile coding Sutton (1996), instead using count-sketch.[1] Ghavamzadeh et al. (2010) investigated sketching features to reduce the dimensionality and make it feasible to run least-squares temporal difference learning (LSTD) for policy evaluation. In LSTD, the value function is estimated by incrementally computing a $d \times d$ matrix $\mathbf{A}$, where d is the number of features, and an d-dimensional vector $\mathbf{b}$, where the parameters are estimated as the solution to this linear system. Because d can be large, they randomly project the features to reduce the matrix size to $k \times k$, with $k \ll d$.

For both of these previous uses of sketching, however, the resulting value function estimates are biased. This bias, as we show in this work, can be quite significant, resulting in significant estimation error in the value function for a given policy. As a result, any gains from us-

[1] They called the sketch the tug-of-war sketch, but it is more standard to call it count-sketch.

ing LSTD methods—over stochastic temporal difference (TD) methods—are largely overcome by this bias. A natural question is if we can benefit from sketching, with minimal bias or without incurring any bias at all.

In this work, we propose to instead sketch the linear system in LSTD. The key idea is to only sketch the constraints of the system (the left-side of $\mathbf{A}$) rather than the variables (the right-side of $\mathbf{A}$). Sketching features, on the other hand, by design, sketches both constraints and variables. We show that even with a straightforward linear system solution, the left-sided sketch can significantly reduce bias. We further show how to use this left-sided sketch within a quasi-Newton algorithm, providing an unbiased policy evaluation algorithm that can still benefit from the computational improvements of sketching.

The key novelty in this work is designing such system-sketching algorithms when also incrementally computing the linear system solution. There is a wealth of literature on sketching linear systems, to reduce computation. In general, however, many sketching approaches cannot be applied to the incremental policy evaluation problem, because the approaches are designed for a static linear system. For example, Gower & Richtárik (2015) provide a host of possible solutions for solving large linear systems. However, they assume access to $\mathbf{A}$ upfront, so the algorithm design, in memory and computation, is not suitable for the incremental setting. Some popular sketching approaches, such as Frequent Directions (Ghashami et al., 2014), has been successfully used for the online setting, for quasi-Newton algorithms (Luo et al., 2016); however, they sketch symmetric matrices, that are growing with number of samples.

This paper is organized as follows. We first introduce the policy evaluation problem—learning a value function for a fixed policy—and provide background on sketching methods. We then illustrate issues with only sketching features, in terms of quality of the value function approximation. We then introduce the idea of using asymmetric sketching for policy evaluation with LSTD, and provide an efficient incremental algorithm that is $O(dk)$ on each step. We finally highlight settings where we expect sketching to perform particularly well in practice, and investigate the properties of our algorithm on four domains, and with a variety of representation properties.

2 PROBLEM FORMULATION

We address the policy evaluation problem within reinforcement learning, where the goal is to estimate the value function for a given policy[2]. As is standard, the agent-environment interaction is formulated as a Markov decision process $(\mathcal{S}, \mathcal{A}, \Pr, r)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $\Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty)$ is the one-step state transition dynamics. On each time step $t = 1, 2, 3, ...$, the agent selects an action according to its policy π, $A_t \sim \pi(S_t, \cdot)$, with $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)$ and transitions into a new state $S_{t+1} \sim \Pr(S_t, A_t, \cdot)$ and obtains scalar reward $R_{t+1} \stackrel{\text{def}}{=} r(S_t, A_t, S_{t+1})$.

For policy evaluation, the goal is to estimate the value function, $v_\pi : \mathcal{S} \rightarrow \mathbb{R}$, which corresponds to the expected return when following policy π

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

where $\mathbb{E}_\pi$ is the expectation over future states when 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:

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

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. A common setting, for example, is a constant discount. This recent generalization to state-dependent discount (Sutton et al., 2011; White, 2016) enables either episodic or continuing problems, and so we adopt this more general formalism here.

We consider linear function approximation to estimate the value function. In this setting, the observations are expanded to a higher-dimensional space, such as through tile-coding, radial basis functions or Fourier basis. Given this nonlinear encoding $x : \mathcal{S} \rightarrow \mathbb{R}^d$, the value is approximated as $v_\pi(S_t) \approx \mathbf{w}^\top \mathbf{x}_t$ for $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{x}_t \stackrel{\text{def}}{=} x(S_t)$.

One algorithm for estimating $\mathbf{w}$ is least-squares temporal difference learning (LSTD). The goal in LSTD(λ) (Boyan, 1999) is to minimize the mean-squared projected Bellman error, which can be represented as solving the following linear system

$$\begin{aligned} \mathbf{A} &\stackrel{\text{def}}{=} \mathbb{E}_\pi[\mathbf{e}_t(\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1})^\top] \\ \mathbf{b} &\stackrel{\text{def}}{=} \mathbb{E}_\pi[R_{t+1}\mathbf{e}_t]. \end{aligned}$$

where $\mathbf{e}_t \stackrel{\text{def}}{=} \gamma_{t+1}\lambda \mathbf{e}_{t-1} + \mathbf{x}_t$ is called the eligibility trace for trace parameter $\lambda \in [0, 1]$. To obtain $\mathbf{w}$, the system $\mathbf{A}$ and $\mathbf{b}$ are incrementally estimated, to solve $\mathbf{A}\mathbf{w} = \mathbf{b}$. For a trajectory $\{(S_t, A_t, S_{t+1}, R_{t+1})\}_{t=0}^{T-1}$,

[2]To focus the investigation on sketching, we consider the simpler on-policy setting in this work. Many of the results, however, generalize to the off-policy setting, where data is generated according to a behavior policy different than the given target policy we wish to evaluate.

let $\mathbf{d}_t \stackrel{\text{def}}{=} \mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1}$, then the above two expected terms are usually computed via sample average that can be recursively computed, in a numerically stable way, as

$$\mathbf{A}_{t+1} = \mathbf{A}_t + \frac{1}{t+1}\left(\mathbf{e}_t\mathbf{d}_t^\top - \mathbf{A}_t\right)$$
$$\mathbf{b}_{t+1} = \mathbf{b}_t + \frac{1}{t+1}\left(\mathbf{e}_t R_{t+1} - \mathbf{b}_t\right)$$

with $\mathbf{A}_0 = \mathbf{0}$ and $\mathbf{b}_0 = \mathbf{0}$. The incremental estimates $\mathbf{A}_t$ and $\mathbf{b}_t$ converge to $\mathbf{A}$ and $\mathbf{b}$. A naive algorithm, where $\mathbf{w} = \mathbf{A}_t^{-1}\mathbf{b}_t$ is recomputed on each step, would result in $\mathcal{O}(d^3)$ computation to compute the inverse $\mathbf{A}_t^{-1}$. Instead, $\mathbf{A}_t^{-1}$ is incrementally updated using the Sherman-Morrison formula, with $\mathbf{A}_0^{-1} = \xi$ for a small $\xi > 0$

$$\mathbf{A}_t^{-1} = \left(\frac{t-1}{t}\mathbf{A}_{t-1} + \frac{1}{t}\mathbf{e}_t\mathbf{d}_t^\top\right)^{-1}$$
$$= \frac{t}{t-1}\left(\mathbf{A}_{t-1}^{-1} + \frac{\mathbf{A}_{t-1}^{-1}\mathbf{e}_t\mathbf{d}_t^\top\mathbf{A}_{t-1}^{-1}}{t-1+\mathbf{d}_t^\top\mathbf{A}_{t-1}^{-1}\mathbf{e}_t}\right)$$

requiring $\mathcal{O}(d^2)$ storage and computation per step. Unfortunately, this quadratic cost is prohibitive for many incremental learning settings. In our experiments, even $d = 10,000$ was prohibitive, since $d^2 = 100$ million.

A natural approach to improve computation to solve for $\mathbf{w}$ is to use stochastic methods, such as TD(λ) (Sutton, 1988). This algorithm incrementally updates with $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\delta_t\mathbf{e}_t$ for stepsize $\alpha > 0$ and TD-error $\delta_t = R_{t+1} + (\gamma_{t+1}\mathbf{x}_{t+1} - \mathbf{x}_t)^\top\mathbf{w}_t$. The expectation of this update is $\mathbb{E}_\pi[\delta_t\mathbf{e}_t] = \mathbf{b} - \mathbf{A}\mathbf{w}_t$; the fixed-point solutions are the same for both LSTD and TD, but LSTD corresponds to a batch solution whereas TD corresponds to a stochastic update. Though more expensive than TD—which is only $O(d)$—LSTD does have several advantages. Because LSTD is a batch method, it summarizes all samples (within $\mathbf{A}$), and so can be more sample efficient. Additionally, LSTD has no step-size parameter, using a closed-form solution for $\mathbf{w}$.

Recently, there has been some progress in better balancing between TD and LSTD. Pan et al. (2017) derived a quasi-Newton algorithm, called accelerated gradient TD (ATD), giving an unbiased algorithm that has some of the benefits of LSTD, but with significantly reduced computation because they only maintain a low-rank approximation to $\mathbf{A}$. The key idea is that $\mathbf{A}$ provides curvature information, and so can significantly improve step-size selection for TD and so improve the convergence rate. The approximate $\mathbf{A}$ can still provide useful curvature information, but can be significantly cheaper to compute. We use the ATD update to similarly obtain an unbiased algorithm, but use sketching approximations instead of low-rank approximations. First, however, we investigate some of the properties of sketching.

3 ISSUES WITH SKETCHING THE FEATURES

One approach to make LSTD more feasible is to project—sketch—the features. Sketching involves sampling a random matrix $\mathbf{S} : \mathbb{R}^{k\times d}$ from a family of matrices $\mathcal{S}$, to project a given d-dimensional vector $\mathbf{x}$ to a (much smaller) k-dimensional vector $\mathbf{S}\mathbf{x}$. The goal in defining this class of sketching matrices is to maintain certain properties of the original vector. The following is a standard definition for such a family.

Definition 1 (Sketching)**.** *Let d and k be positive integers, $\delta \in (0,1)$, and $\epsilon \in \mathbb{R}^+$. Then, $\mathcal{S} \subset \mathbb{R}^{k\times d}$ is called a family of sketching matrices with parameters (ϵ, δ), if for a random matrix, $\mathbf{S}$, chosen uniformly at random from this family, we have that $\forall \mathbf{x} \in \mathbb{R}^d$*

$$\mathbb{P}\Big[(1-\epsilon)\|\mathbf{x}\|_2^2 \le \|\mathbf{S}\mathbf{x}\|_2^2 \le (1+\epsilon)\|\mathbf{x}\|_2^2\Big] \ge 1-\delta.$$

where the probability is w.r.t. the distribution over $\mathbf{S}$.

We will explore the utility of sketching the features with several common sketches. These sketches all require $k = \Omega(\epsilon^{-2}\ln(1/\delta)\ln d)$.

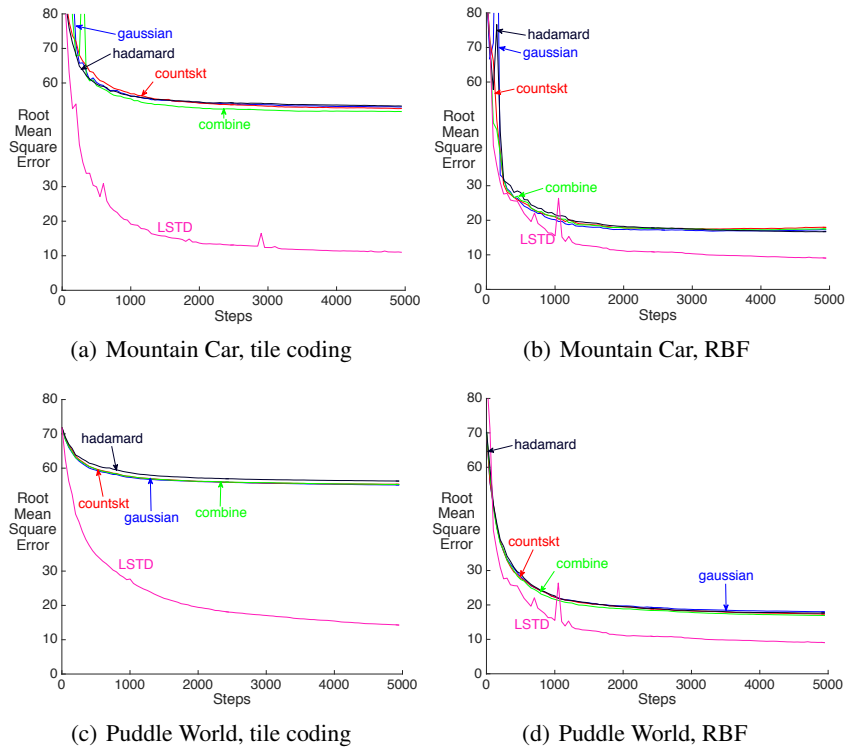
Gaussian random projections, also known as the JL-Transform (Johnson & Lindenstrauss, 1984), has each entry in $\mathbf{S}$ i.i.d. sampled from a Gaussian, $\mathcal{N}(0, \frac{1}{k})$.

Count sketch selects exactly one uniformly picked non-zero entry in each column, and sets that entry to either 1 or -1 with equal probability (Charikar et al., 2002; Gilbert & Indyk, 2010). The Tug-of-War sketch (Alon et al., 1996) performs very similarly to Count sketch in our experiments, and so we omit it.

Combined sketch is the product of a count sketch matrix and a Gaussian projection matrix (Wang, 2015; Boutsidis & Woodruff, 2015).

Hadamard sketch—the Subsampled Randomized Hadamard Transform—is computed as $\mathbf{S} = \frac{1}{\sqrt{kd}}\mathbf{D}\mathbf{H}_d\mathbf{P}$, where $\mathbf{D} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with each diagonal element uniformly sampled from $\{1, -1\}$, $\mathbf{H}_d \in \mathbb{R}^{d\times d}$ is a Hadamard matrix and $\mathbf{P} \in \mathbb{R}^{d\times k}$ is a column sampling matrix (Ailon & Chazelle, 2006).

Sketching provides a low-error between the recovery $\mathbf{S}^\top\mathbf{S}\mathbf{x}$ and the original $\mathbf{x}$, with high probability. For the above families, the entries in $\mathbf{S}$ are zero-mean i.i.d. with variance 1, giving $\mathbb{E}[\mathbf{S}^\top\mathbf{S}] = \mathbf{I}$ over all possible $\mathbf{S}$. Consequently, in expectation, the recovery $\mathbf{S}^\top\mathbf{S}\mathbf{x}$ is equal to $\mathbf{x}$. For a stronger result, a Chernoff bound can be used to bound the deviation of $\mathbf{S}^\top\mathbf{S}$ from this expected value: for the parameters (ϵ, δ) of the matrix family, we get that $\mathbb{P}\Big[(1-\epsilon)I \prec \mathbf{S}^\top\mathbf{S} \prec (1+\epsilon)I\Big] \ge 1-\delta$.



(a) Mountain Car, tile coding

(b) Mountain Car, RBF

(c) Puddle World, tile coding

(d) Puddle World, RBF

Figure 1: Efficacy of different sketches for sketching the features for LSTD, with $k = 50$. The RMSE is w.r.t. the optimal value function, computed using rollouts. LSTD(λ) is included as the baseline, with $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$, with the other curves corresponding to different sketches of the features, to give $\mathbf{w} = (\mathbf{SAS}^\top)^{-1}\mathbf{Sb}$ as used for the random projections LSTD algorithm. The RBF width in Mountain Car is $\sigma = 0.12$ times the range of the state space and in Puddle World is $\sigma = \sqrt{0.0072}$. The 1024 centers for RBFs are chosen to uniformly cover the 2-d space in a grid. For tile coding, we discretize each dimension by 10, giving 10×10 grids, use 10 tilings, and set the memory size as 1024. The bias is high for tile coding features, and much better for RBF features, though still quite large. The different sketches perform similarly.

These properties suggest that using sketching for the feature vectors should provide effective approximations. Bellemare et al. (2012) showed that they could use these projections for tile coding, rather than the biased hashing function that is typically used, to improve learning performance for the control setting. The efficacy, however, of sketching given features, versus using the unsketched features, is less well-understood.

We investigate the properties of sketching the features, shown in Figure 1 with a variety of sketches in two benchmark domains for RBF and tile-coding representations (see (Sutton & Barto, 1998, Chapter 8) for an overview of these representations). For both domains, the observations space is 2-dimensional, with expansion to $d = 1024$ and $k = 50$. The results are averaged over 50 runs, with ξ, λ swept over 13 values, with ranges listed in Appendix C. We see that sketching the features can incur significant bias, particularly for tile coding, even with a reasonably large $k = 50$ to give $O(dk)$ runtimes. This bias reduces with k, but remains quite high and so is likely too unreliable for practical use.

4 SKETCHING THE LINEAR SYSTEM

All of the work on sketching within reinforcement learning has investigated sketching the features; however, we can instead consider sketching the linear system, $\mathbf{Aw} = \mathbf{b}$. For such a setting, we can sketch the left and right subspaces of $\mathbf{A}$ with different sketching matrices, $\mathbf{S}_L \in \mathbb{R}^{k_L \times d}$ and $\mathbf{S}_R \in \mathbb{R}^{k_R \times d}$. Depending on the choices of k_L and k_R, we can then solve the smaller system $\mathbf{S}_L\mathbf{A}\mathbf{S}_R^\top\mathbf{S}_R\mathbf{w} = \mathbf{S}_L\mathbf{b}$ efficiently. The goal is to better take advantage of the properties for the different sides of an asymmetric matrix $\mathbf{A}$.

One such natural improvement should be in one-sided sketching. By only sketching from the left, for example, and setting $\mathbf{S}_R = \mathbf{I}$, we do not project $\mathbf{w}$. Rather, we only project the constraints to the linear system $\mathbf{Aw} = \mathbf{b}$. Importantly, this does not introduce bias: the original solution $\mathbf{w}$ to $\mathbf{Aw} = \mathbf{b}$ is also a solution to $\mathbf{SAw} = \mathbf{Sb}$ for any sketching matrix $\mathbf{S}$. The projection, however, removes uniqueness in terms of the solutions $\mathbf{w}$, since the system is under-constrained. Conversely, by only sketching from the right, and setting $\mathbf{S}_L = \mathbf{I}$, we constrain the space of solutions to a unique set, and do not remove any constraints. For this setting, however, it is unlikely that $\mathbf{w}$ with $\mathbf{Aw} = \mathbf{b}$ satisfies $\mathbf{AS}^\top\mathbf{w} = \mathbf{b}$.

The conclusion from many initial experiments is that the key benefit from asymmetric sketching is when only sketching from the left. We experimented with all pairwise combinations of Gaussian random projections, Count sketch, Tug-of-War sketch and Hadamard sketch for $\mathbf{S}_L$ and $\mathbf{S}_R$. We additionally experimented with only sketching from the right, setting $\mathbf{S}_L = \mathbf{I}$. In all of these experiments, we found asymmetric sketching provided little to no benefit over using $\mathbf{S}_L = \mathbf{S}_R$ and that sketch-

ing only from the right also performed similarly to using $\mathbf{S}_L = \mathbf{S}_R$. We further investigated column and row selection sketches (see Wang (2015) for a thorough overview), but also found these to be ineffective. We therefore proceed with an investigation into effectively using left-side sketching. In the next section, we provide an efficient $\mathcal{O}(dk)$ algorithm to compute $(\mathbf{S}_L\mathbf{A})^\dagger$, to enable computation of $\mathbf{w} = (\mathbf{S}_L\mathbf{A})^\dagger\mathbf{S}_L\mathbf{b}$ and for use within an unbiased quasi-Newton algorithm.

We conclude this section with an interesting connection to a data-dependent projection method that has been used for policy evaluation, that further motivates the utility of sketching only from the left. This algorithm, called truncated LSTD (tLSTD) (Gehring et al., 2016), incrementally maintains a rank k approximation of $\mathbf{A}$ matrix, using an incremental singular value decomposition. We show below that this approach corresponds to projecting $\mathbf{A}$ from the left with the top k left singular vectors. This is called a data-dependent projection, because the projection depends on the observed data, as opposed to the data-independent projection—the sketching matrices—which is randomly sampled independently of the data.

Proposition 1. *Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be singular value decomposition of the true $\mathbf{A}$. Assume the singular values are in decreasing order and let $\mathbf{\Sigma}_k$ be the top k singular values, with corresponding k left singular vectors $\mathbf{U}_k$ and k right singular vectors $\mathbf{V}_k$. Then the solution $\mathbf{w} = \mathbf{V}_k\mathbf{\Sigma}_k^\dagger\mathbf{U}_k^\top\mathbf{b}$ (used for tLSTD) corresponds to LSTD using asymmetric sketching with $\mathbf{S}_L = \mathbf{U}_k^\top$ and $\mathbf{S}_R = \mathbf{I}$.*

Proof. We know $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$ for singular vectors $\mathbf{u}_i \in \mathbb{R}^d$ with $\mathbf{u}_i^\top\mathbf{u}_i = 1$ and $\mathbf{u}_i^\top\mathbf{u}_j = 0$ for $i \neq j$. Since $\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$, we get that $\mathbf{U}_k^\top\mathbf{U} = [\mathbf{I}_k \;\; \mathbf{0}_{d-k}] \in \mathbb{R}^{k\times d}$ for k-dimensional identity matrix $\mathbf{I}_k$ and zero matrix $\mathbf{0}_{d-k} \in \mathbb{R}^{k\times(d-k)}$. Then we get that $\mathbf{S}_L\mathbf{b} = \mathbf{U}_k^\top\mathbf{b}$ and $\mathbf{S}_L\mathbf{A} = [\mathbf{I}_k \;\; \mathbf{0}_{d-k}]\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{\Sigma}_k\mathbf{V}^\top = \mathbf{\Sigma}_k\mathbf{V}_k^\top$. Therefore, $\mathbf{w} = (\mathbf{S}_L\mathbf{A})^\dagger\mathbf{S}_L\mathbf{b} = \mathbf{V}_k\mathbf{\Sigma}_k^\dagger\mathbf{U}_k^\top\mathbf{b}$. □

5 LEFT-SIDED SKETCHING ALGORITHM

In this section, we develop an efficient approach to use the smaller, sketched matrix $\mathbf{SA}$ for incremental policy evaluation. The most straightforward way to use $\mathbf{SA}$ is to incrementally compute $\mathbf{SA}$, and periodically solve $\mathbf{w} = (\mathbf{SA})^\dagger\mathbf{Sb}$. This costs $O(dk)$ per step, and $O(d^2k)$ every time the solution is recomputed. To maintain $O(dk)$ computation per-step, this full solution could only be computed every d steps, which is too infrequent to provide a practical incremental policy evaluation approach. Further, because it is an underconstrained system, there are likely to be infinitely many solutions to $\mathbf{SAw} = \mathbf{Sb}$; amongst those solutions, we would like to sub-select amongst the unbiased solutions to $\mathbf{Aw} = \mathbf{b}$.

We first discuss how to efficiently maintain $(\mathbf{SA})^\dagger$, and then describe how to use that matrix to obtain an unbiased algorithm. Let $\tilde{\mathbf{A}} \stackrel{\text{def}}{=} \mathbf{SA} \in \mathbb{R}^{k\times d}$. For this underconstrained system with $\tilde{\mathbf{b}} \stackrel{\text{def}}{=} \mathbf{Sb}$, the minimum norm solution to $\tilde{\mathbf{A}}\mathbf{w} = \tilde{\mathbf{b}}$ is[3] $\mathbf{w} = \tilde{\mathbf{A}}^\top(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^\top)^{-1}\tilde{\mathbf{b}}$ and $\tilde{\mathbf{A}}^\dagger = \tilde{\mathbf{A}}^\top(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^\top)^{-1} \in \mathbb{R}^{d\times k}$. To maintain $\tilde{\mathbf{A}}_t^\dagger$ incrementally, therefore, we simply need to maintain $\tilde{\mathbf{A}}_t$ incrementally and the $k \times k$-matrix $(\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top)^{-1}$ incrementally. Let $\tilde{\mathbf{e}}_t \stackrel{\text{def}}{=} \mathbf{S}\mathbf{e}_t$, $\mathbf{d}_t \stackrel{\text{def}}{=} \mathbf{d}_t$ and $\mathbf{h}_t \stackrel{\text{def}}{=} \tilde{\mathbf{A}}_t\mathbf{d}_t$. We can update the sketched system in $O(dk)$ time and space

$$\tilde{\mathbf{A}}_{t+1} = \tilde{\mathbf{A}}_t + \tfrac{1}{t+1}\left(\tilde{\mathbf{e}}_t\mathbf{d}_t^\top - \tilde{\mathbf{A}}_t\right)$$
$$\tilde{\mathbf{b}}_{t+1} = \tilde{\mathbf{b}}_t + \tfrac{1}{t+1}\left(\tilde{\mathbf{e}}_t R_{t+1} - \tilde{\mathbf{b}}_t\right)$$

To maintain $(\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top)^{-1}$ incrementally, notice that the unnormalized update is

$$\begin{aligned}\tilde{\mathbf{A}}_{t+1}\tilde{\mathbf{A}}_{t+1}^\top &= (\tilde{\mathbf{A}}_t + \tilde{\mathbf{e}}_t\mathbf{d}_t^\top)(\tilde{\mathbf{A}}_t + \tilde{\mathbf{e}}_t\mathbf{d}_t^\top) \\ &= \tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top + \tilde{\mathbf{e}}_t\mathbf{h}_t^\top + \mathbf{h}_t\tilde{\mathbf{e}}_t^\top + ||\mathbf{d}_t||_2^2\|\tilde{\mathbf{e}}_t\tilde{\mathbf{e}}_t^\top.\end{aligned}$$

Hence, $(\tilde{\mathbf{A}}_{t+1}\tilde{\mathbf{A}}_{t+1}^\top)^{-1}$ can be updated from $(\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top)^{-1}$ by applying the Sherman-Morrison update three times. For a normalized update, based on samples, the update is

$$\begin{aligned}\tilde{\mathbf{A}}_{t+1}\tilde{\mathbf{A}}_{t+1}^\top = &\left(\tfrac{t}{t+1}\right)^2\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top + \tfrac{t}{(t+1)^2}\left(\tilde{\mathbf{e}}_t\mathbf{h}_t^\top + \mathbf{h}_t\tilde{\mathbf{e}}_t^\top\right) \\ &+ \tfrac{1}{(t+1)^2}||\mathbf{d}_t||_2^2\|\tilde{\mathbf{e}}_t\tilde{\mathbf{e}}_t^\top\end{aligned}$$

We can then compute $\mathbf{w}_t = \tilde{\mathbf{A}}_t(\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top)^\dagger\tilde{\mathbf{b}}_t$ on each step.

This solution, however, will provide the minimum norm solution, rather than the unbiased solution, even though the unbiased solution is feasible for the underconstrained system. To instead push the preference towards this unbiased solution, we use the stochastic approximation algorithm, called ATD (Pan et al., 2017). This method is a quasi-second order method, that relies on a low-rank approximation $\hat{\mathbf{A}}_t$ to $\mathbf{A}_t$; using this approximation, the update is $\mathbf{w}_{t+1} = \mathbf{w}_t + (\alpha_t\hat{\mathbf{A}}_t^\dagger + \eta\mathbf{I})\delta_t\mathbf{e}_t$. Instead of being used to explicitly solve for $\mathbf{w}$, the approximation matrix is used to provide curvature information. The inclusion of η constitutes a small regularization component, that pushes the solution towards the unbiased solution.

We show in the next proposition that for our alternative approximation, we still obtain unbiased solutions. We use results for iterative methods for singular linear systems (Shi et al., 2011; Wang & Bertsekas, 2013), since

[3] We show in Proposition 2, Appendix A, that $\tilde{\mathbf{A}}$ is full row rank with high probability. This property is required to ensure that the inverse of $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^\top$ exists. In practice, this is less of a concern, because we initialize the matrix $\tilde{\mathbf{A}}_0\tilde{\mathbf{A}}_0^\top$ with a small positive value, ensuring invertibility for $\tilde{\mathbf{A}}_t\tilde{\mathbf{A}}_t^\top$ for finite t.

$\mathbf{A}$ may be singular. $\mathbf{A}$ has been shown to be positive semi-definite under standard assumptions on the MDP (Yu, 2015); for simplicity, we assume $\mathbf{A}$ is positive semi-definite, instead of providing these MDP assumptions.

Assumption 1. *For* $\mathbf{S} \in \mathbb{R}^{k \times d}$ *and* $\mathbf{B} = \alpha(\mathbf{SA})^\dagger \mathbf{S} + \eta \mathbf{I}$ *with* $\mathbf{B} \in \mathbb{R}^{d \times d}$*, the matrix* $\mathbf{BA}$ *is diagonalizable.*

Assumption 2. $\mathbf{A}$ *is positive semi-definite.*

Assumption 3. $\alpha \in (0, \frac{1}{2})$ *and* $0 < \eta \leq \frac{1}{2\lambda_{max}(\mathbf{A})}$ *where* $\lambda_{max}(\mathbf{A})$ *is the maximum eigenvalue of* $\mathbf{A}$.

Theorem 1. *Under Assumptions 1-3, the expected updating rule* $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbb{E}_\pi[\mathbf{B}\delta_t \mathbf{e}_t]$ *converges to a fixed-point* $\mathbf{w}^\star = \mathbf{A}^\dagger \mathbf{b}$.

Proof. The expected updating rule is $\mathbb{E}_\pi[\mathbf{B}\delta_t \mathbf{e}_t] = \mathbf{B}(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$. As in the proof of convergence for ATD (Pan et al., 2017, Theorem 1), we similarly verify the conditions from (Shi et al., 2011, Theorem 1.1).

Notice first that $\mathbf{BA} = \alpha(\mathbf{SA})^\dagger \mathbf{SA} + \eta \mathbf{A}$.

For singular value decomposition, $\mathbf{SA} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, we have that $(\mathbf{SA})^\dagger \mathbf{SA} = \mathbf{V}\mathbf{\Sigma}^\dagger \mathbf{U}^\top \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{V}[\mathbf{I}_{\tilde{k}}\ \mathbf{0}_{d-k}]\mathbf{V}^\top$, where $\tilde{k} \leq k$ is the rank of $\mathbf{SA}$. The maximum eigenvalue of $(\mathbf{SA})^\dagger \mathbf{SA}$ is therefore 1.

Because $(\mathbf{SA})^\dagger \mathbf{SA}$ and $\mathbf{A}$ are both positive semidefinite, $\mathbf{BA}$ is positive semi-definite. By Weyl's inequalities,

$$\lambda_{\max}(\mathbf{BA}) \leq \alpha \lambda_{\max}((\mathbf{SA})^\dagger \mathbf{SA}) + \eta \lambda_{\max}(\mathbf{A}).$$

Therefore, the eigenvalues of $\mathbf{I} - \mathbf{BA}$ have absolute value strictly less than 1, because $\eta \leq (2\lambda_{\max}(\mathbf{A}))^{-1}$ and $\alpha < 1/2 = (2\lambda_{\max}((\mathbf{SA})^\dagger \mathbf{SA}))^{-1}$ by assumption.

For the second condition, since $\mathbf{BA}$ is PSD and diagonalizable, we can write $\mathbf{BA} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$ for some matrices $\mathbf{Q}$ and diagonal matrix $\mathbf{\Lambda}$ with eigenvalues greater than or equal to zero. Then $(\mathbf{BA})^2 = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} = \mathbf{Q}\mathbf{\Lambda}^2\mathbf{Q}^{-1}$ has the same rank.

For the third condition, because $\mathbf{BA}$ is the sum of two positive semi-definite matrices, the nullspace of $\mathbf{BA}$ is a subset of the nullspace of each of those matrices individually: nullspace$(\mathbf{BA})$ = nullspace$(\alpha(\mathbf{SA})^\dagger \mathbf{SA} + \eta \mathbf{A}) \subseteq \eta$nullspace$(\eta \mathbf{A})$ = nullspace$(\mathbf{A})$. In the other direction, for all $\mathbf{w}$ such that $\mathbf{A}\mathbf{w} = \mathbf{0}$, its clear that $\mathbf{BA}\mathbf{w} = \mathbf{0}$, and so nullspace$(\mathbf{A}) \subseteq$ nullspace$(\mathbf{BA})$. Therefore, nullspace$(\mathbf{A})$ = nullspace$(\mathbf{BA})$. □

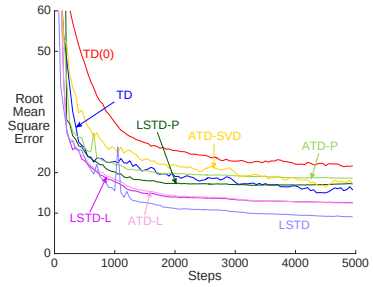
6 WHEN SHOULD SKETCHING HELP?

To investigate the properties of these sketching approaches, we need to understand when we expect sketching to have the most benefit. Despite the wealth of literature on sketching and strong theoretical results, there seems to be fewer empirical investigations into when sketching has most benefit. In this section, we elucidate some hypotheses about when sketching should be most effective, which we then explore in our experiments.

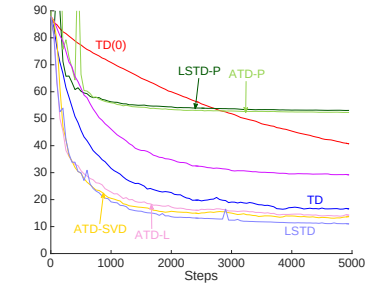
In the experiments for sketching the features in Section 3, it was clear that sketching the RBF features was much more effective than sketching the tile coding features. A natural investigation, therefore, is into the properties of representations that are more amenable to sketching. The key differences between these two representations is in terms of smoothness, density and overlap. The tile coding representation has non-smooth 0,1 features, which do not overlap in each grid. Rather, the overlap for tile coding results from overlapping tilings. This differs from RBF overlap, where centers are arranged in a grid and only edges of the RBF features overlap. The density of RBF features is significantly higher, since more RBFs are active for each input. Theoretical work in sketching for regression (Maillard & Munos, 2012), however, does not require features to be smooth. We empirically investigate these three properties—smoothness, density and overlap.

There are also some theoretical results that suggest sketching could be more amenable for more distinct features—less overlap or potentially less tilings. Balcan et al. (2006) showed a worst-case setting where data-independent sketching results in poor performance. They propose a two-stage projection, to maintain separability in classification. The first stage uses a data-dependent projection, to ensure features are not highly correlated, and the second uses a data-independent projection (a sketch) to further reduce the dimensionality after the orthogonal projection. The implied conclusion from this result is that, if the features are not highly correlated, then the first step can be avoided and the data independent sketch should similarly maintain classification accuracy. This result suggests that sketching for feature expansions with less redundancy should perform better.

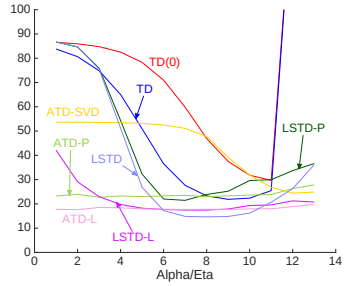
We might also expect sketching to be more robust to the condition number of the matrix. For sketching in regression, Fard et al. (2012) found a bias-variance tradeoff when increasing k, where for large k, estimation error from a larger number of parameters became a factor. Similarly, in our experiments above, LSTD using an incremental Sherman-Morrison update has periodic spikes in the learning curve, indicating some instability. The smallest eigenvalue of the sketched matrix should be larger than that of the original matrix; this improvement in condition number compensates for the loss in information. Similarly, we might expect that maintaining an incremental singular value decomposition, for ATD, could be less robust than ATD with left-side sketching.



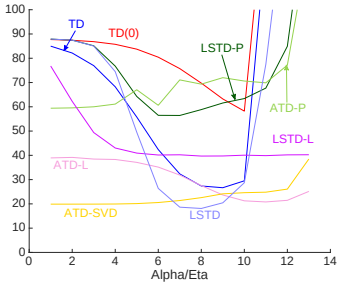
(a) Mountain Car, RBF



(b) Mountain Car, Tile coding

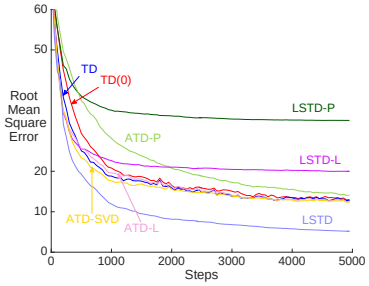


(c) Mountain Car, RBF, Sensitivity

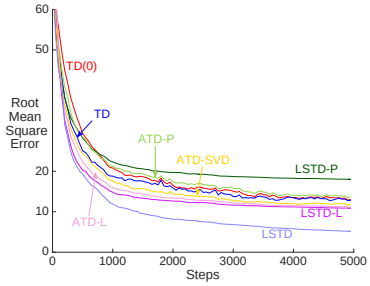


(d) Mountain Car, Tile coding, Sensitivity

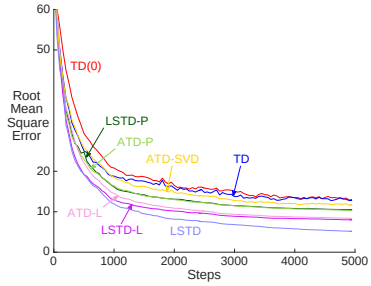
Figure 2: **(a)** and **(b)** are learning curves on Mountain Car with $k = 50$, and **(c)** and **(d)** are their corresponding parameter-sensitivity plots. The sensitivity plots report average RMSE over the entire learning curve, for the best λ for each parameter. The stepsize α is reported for TD, the initialization parameter ξ for the LSTD methods and the regularization parameter η for the ATD methods. The initialization for the matrices in the ATD methods is fixed to the identity. The range for the regularization term η is 0.1 times the range for α. As before, the sketching approaches with RBFs perform better than with tile coding. The sensitivity of the left-side projection methods is significantly lower than the TD methods. ATD-L also seems to be less sensitive than ATD-SVD, and incurs less bias than LSTD-L.



(a) Puddle World, RBF, $k = 25$



(b) Puddle World, RBF, $k = 50$



(c) Puddle World, RBF, $k = 75$

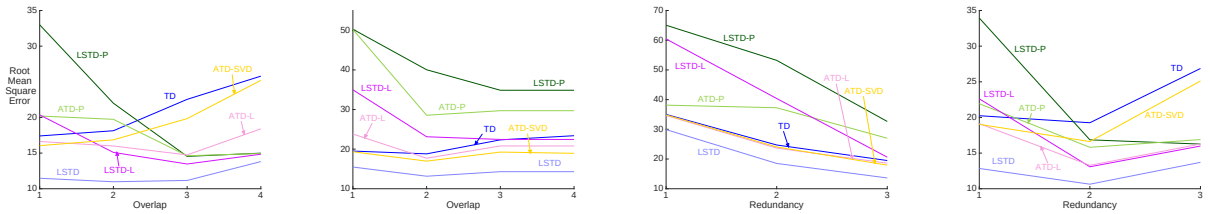
Figure 3: Change in performance when increasing k, from 25 to 75. Two-sided projection (i.e., projecting the features) significantly improves with larger k, but is strictly dominated by left-side projection. At $k = 50$, the left-side projection methods are outperforming TD and are less variant. ATD-SVD seems to gain less with increasing k, though in general we found ATD-SVD to perform more poorly than ATD-P particularly for RBF representations.

7 EXPERIMENTS

In this section, we test the efficacy of sketching for LSTD and ATD in four domains: Mountain Car, Puddle World, Acrobot and Energy Allocation. We set $k = 50$, unless otherwise specified, average all results over 50 runs and sweep parameters for each algorithm. Detailed experimental settings, such as parameter ranges, are in Appendix C. To distinguish projections, we add -P for two-sided and -L for left-sided to the algorithm name.

We conclude that 1) two-sided projection—projecting the features—generally does much worse than only projecting the left-side of $\mathbf{A}$, 2) higher feature density is more amenable to sketching, particularly for two-sided sketching, 3) smoothness of features only seems to impact two-sided sketching, 4) ATD with sketching decreases bias relative to its LSTD variant and 5) ATD with left-sided sketching typically performs as well as ATD-SVD, but is significantly faster.

Performance and parameter sensitivity for RBFs and Tile coding. We first more exhaustively compare the algorithms in Mountain Car and Puddle World, in Figures 2 and 3 with additional such results in the appendix. As has been previously observed, TD with a well-chosen stepsize can perform almost as well as LSTD in terms of sample efficiency, but is quite sensitive to the stepsize.



(a) RBF, $k = 50$ (b) Spline, $k = 50$ (c) Tile coding, $k = 50$ (d) RBFs with tilings, $k = 50$

Figure 4: The effect of varying the representation properties, in Puddle World with $d = 1024$. In **(a)** and **(b)**, we examine the impact of varying the overlap, for both smooth features (RBFs) and 0-1 features (Spline). For spline, the feature is 1 if $||\mathbf{x} - \mathbf{c}_i|| < \sigma$ and otherwise 0. The spline feature represents a bin, like for tile coding, but here we adjust the widths of the bins so that they can overlap and do not use tilings. The x-axis has four width values, to give a corresponding feature vector norm of about $20, 40, 80, 120$. In **(c)** and **(d)**, we vary the redundancy, where number of tilings is increased and the total number of features kept constant. We generate tilings for RBFs like for tile coding, but for each grid cell use an RBF similarity rather than a spline similarity. We used $4 \times 16 \times 16, 16 \times 8 \times 8$ and $64 \times 4 \times 4$.

Here, therefore, we explore if our matrix-based learning algorithms can reduce this parameter sensitivity. In Figure 2, we can indeed see that this is the case. The LSTD algorithms look a bit more sensitive, because we sweep over small initialization values for completeness. For tile coding, the range is a bit more narrow, but for RBFs, in the slightly larger range, the LSTD algorithms are quite insensitive for RBFs. Interestingly, LSTD-L seems to be more robust. We hypothesize that the reason for this is that LSTD-L only has to initialize a smaller $k \times k$ symmetric matrix, $(\mathbf{SA}(\mathbf{SA})^\top)^{-1} = \eta\mathbf{I}$, and so is much more robust to this initialization. In fact, across settings, we found initializing to $\mathbf{I}$ was effective. Similarly, ATD-L benefits from this robustness, since it needs to initialize the same matrix, and then further overcomes bias using the approximation to $\mathbf{A}$ only for curvature information.

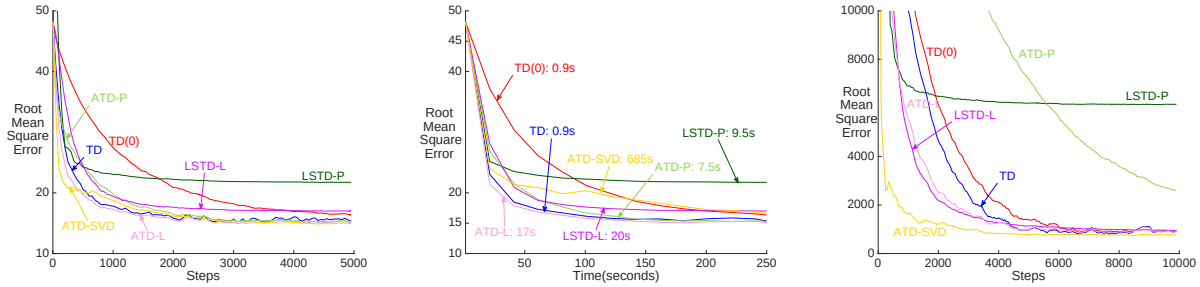
Impact of the feature properties. We explored the feature properties—smoothness, density, overlap and redundancy—where we hypothesized sketching should help, shown in Figure 4. The general conclusions are 1) the two-side sketching methods improve—relative to LSTD—with increasing density (i.e., increasing overlap and increasing redundancy), 2) the smoothness of the features (RBF versus spline) seems to affect the two-side projection methods much more, 3) the shape of the left-side projection methods follows that of LSTD and 4) ATD-SVD appears to follow the shape of TD more. Increased density generally seemed to degrade TD, and so ATD-SVD similarly suffered more in these settings. In general, the ATD methods had less gain over their corresponding LSTD variants, with increasing density.

Experiments on high dimensional domains. We finally apply our sketching techniques on two high dimensional domains to illustrate practical usability: Acrobot and Energy allocation. The Acrobot domain (Sutton & Barto, 1998) is a four dimensional episodic task, where the goal is to raise an arm to a certain height. The Energy allocation domain Salas & Powell (2013) is a five-dimensional continuing task, where the goal is to store and allocate energy to maximize profit. For Acrobot, we used $14,400$ uniformly-spaced centers and for Energy allocation, we used the same tile coding of 8192 features as Pan et al. (2017). We summarize the results in the caption of Figure 5, with the overall conclusion that ATD-L provides an attractive way to reduce parameter sensitivity of TD, and benefit from sketching to reduce computation.

8 CONCLUSION AND DISCUSSION

In this work, we investigated how to benefit from sketching approaches for incremental policy evaluation. We first showed that sketching features can have significant bias issues, and proposed to instead sketch the linear system, enabling better control over how much information is lost. We highlighted that sketching for radial basis features seems to be much more effective, than for tile coding, and further that a variety of natural asymmetric sketching approaches for sketching the linear system are not effective. We then showed that more carefully using sketching—particularly with left-side sketching within a quasi-Newton update—enables us to obtain an unbiased approach that can improve sample efficiency without incurring significant computation. Our goal in this work was to provide practical methods that can benefit from sketching, and start a focus on empirically investigating settings in which sketching is effective.

Sketching has been used for quasi-Newton updates in online learning; a natural question is if those methods are applicable for policy evaluation. Luo et al. (2016) consider sketching approaches for an online Newton-update, for general functions rather than just the linear func-



(a) Acrobot, RBF (b) Acrobot, RMSE vs Time (c) Energy allocation, Tile coding

Figure 5: Results in domains with high-dimensional features, using $k = 50$ and with results averaged over 30 runs. For Acrobot, the (left-side) sketching methods perform well and are much less sensitive to parameters than TD. For runtime, we show RMSE versus time allowing the algorithms to process up to 25 samples per second, to simulate a real-time setting learning; slow algorithms cannot process all 25 within a second. With computation taken into account, ATD-L has a bigger win over ATD-SVD, and does not lose relative to TD. Total runtime in seconds for one run for each algorithm is labeled in the plot. ATD-SVD is much slower, because of the incremental SVD. For the Energy Allocation domain, the two-side projection methods (LSTD-P, ATD-P) are significantly worse than other algorithms. Interestingly, here ATD-SVD has a bigger advantage, likely because sketching the tile coding features is less effective.

tion approximation case we consider here. They similarly have to consider updates amenable to incrementally approximating a matrix (a Hessian in their case). In general, however, porting these quasi-Newton updates to policy evaluation for reinforcement learning is problematic for two reasons. First, the objective function for temporal difference learning is the mean-squared projected Bellman error, which is the product of three expectations. It is not straightforward to obtain an unbiased sample of this gradient, which is why Pan et al. (2017) propose a slightly different quasi-Newton update that uses $\mathbf{A}$ as a preconditioner. Consequently, it is not straightforward to apply quasi-Newton online algorithms that assume access to unbiased gradients. Second, the Hessian can be nicely approximated in terms of gradients, and is symmetric; both are exploited when deriving the sketched online Newton-update (Luo et al., 2016). We, on the other hand, have an asymmetric matrix $\mathbf{A}$.

In the other direction, we could consider if our approach could be beneficial for the online regression setting. For linear regression, with $\gamma = 0$, the matrix $\mathbf{A}$ actually corresponds to the Hessian. In contrast to previous approaches that sketched the features (Maillard & Munos, 2012; Fard et al., 2012; Luo et al., 2016), therefore, one could instead sketch the system and maintain $(\mathbf{SA})^\dagger$. Since the second-order update is $\mathbf{A}^{-1}\mathbf{g}_t$ for gradient $\mathbf{g}_t$ on iteration t, an approximate second-order update could be computed as $((\mathbf{SA})^\dagger\mathbf{S} + \eta\mathbf{I})\mathbf{g}_t$.

In our experiments, we found sketching both sides of $\mathbf{A}$ to be less effective and found little benefit from modifying the chosen sketch; however, these empirical conclusions warrant further investigation. With more understanding into the properties of $\mathbf{A}$, it could be possible to benefit from this variety. For example, sketching the left-side of $\mathbf{A}$ could be seen as sketching the eligibility trace, and the right-side as sketching the difference between successive features. For some settings, there could be properties of either of these vectors that are particularly suited to a certain sketch. As another example, the key benefit of many of the sketches over Gaussian random projections is in enabling the dimension k to be larger, by using (sparse) sketching matrices where dot product are efficient. We could not easily benefit from these properties, because $\mathbf{SA}$ could be dense and computing matrix-vector products and incremental inverses would be expensive for larger k. For sparse $\mathbf{A}$, or when $\mathbf{SA}$ has specialized properties, it could be more possible to benefit from different sketches.

Finally, the idea of sketching fits well into a larger theme of random representations within reinforcement learning. A seminal paper on random representations (Sutton & Whitehead, 1993) demonstrates the utility of random threshold units, as opposed to more carefully learned units. Though end-to-end training has become more popular in recent years, there is evidence that random representations can be quite powerful (Aubry & Jaffard, 2002; Rahimi & Recht, 2007, 2008; Maillard & Munos, 2012), or even combined with descent strategies (Mahmood & Sutton, 2013). For reinforcement learning, this learning paradigm is particularly suitable, because data cannot be observed upfront. Data-independent representations, such as random representations and sketching approaches, are therefore particularly appealing and warrant further investigation for the incremental learning setting within reinforcement learning.

References

Ailon, Nir and Chazelle, Bernard. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. *ACM Symposium on Theory of Computing*, 2006.

Alon, Noga, Matias, Yossi, and Szegedy, Mario. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, 1996.

Aubry, Jean-Marie and Jaffard, Stéphane. Random Wavelet Series. *Communications in Mathematical Physics*, 2002.

Balcan, Maria-Florina, Blum, Avrim, and Vempala, Santosh. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 2006.

Bellemare, Marc, Veness, Joel, and Bowling, Michael. Sketch-Based Linear Value Function Approximation. In *Advances in Neural Information Processing Systems*, 2012.

Boutsidis, Christos and Woodruff, David P. Communication-optimal distributed principal component analysis in the column-partition model. *arXiv:1504.06729*, 2015.

Boyan, J and Moore, A W. Generalization in Reinforcement Learning: Safely Approximating the Value Function. *Advances in Neural Information Processing Systems*, 1995.

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

Charikar, M, Chen, K, and Farach-Colton, M. Finding frequent items in data streams. *Theoretical Computer Science*, 2002.

Fard, Mahdi Milani, Grinberg, Yuri, Pineau, Joelle, and Precup, Doina. Compressed Least-Squares Regression on Sparse Spaces. *AAAI*, 2012.

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

Ghashami, M, Desai, A, and Phillips, J M. Improved practical matrix sketching with guarantees. *European Symposium on Algorithms*, 2014.

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

Gilbert, A and Indyk, P. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 2010.

Gower, Robert M and Richtárik, Peter. Randomized Iterative Methods for Linear Systems. *arXiv:1506.03296*, 2015.

Johnson, William B and Lindenstrauss, Joram. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 1984.

Luo, Haipeng, Agarwal, Alekh, Cesa-Bianchi, Nicolò, and Langford, John. Efficient Second Order Online Learning by Sketching. *Advances in Neural Information Processing Systems*, 2016.

Mahmood, A R and Sutton, R.S. Representation search through generate and test. In *Proceedings of the AAAI Workshop on Learning Rich Representations from Low-Level Sensors*, 2013.

Maillard, Odalric-Ambrym and Munos, Rémi. Linear Regression With Random Projections. *Journal of Machine Learning Research*, 2012.

Pan, Yangchen, White, Adam, and White, Martha. Accelerated Gradient Temporal Difference Learning. In *AAAI Conference on Artifical Intelligence*, 2017.

Rahimi, A and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

Rahimi, Ali and Recht, Benjamin. Uniform approximation of functions with random bases. In *Annual Allerton Conference on Communication, Control, and Computing*, 2008.

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

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

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

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

Sutton, R.S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, 1996.

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

Sutton, RS and Whitehead, SD. Online learning with random representations. In *International Conference on Machine Learning*, 1993.

Wang, Mengdi and Bertsekas, Dimitri P. On the convergence of simulation-based iterative methods for solving singular linear systems. *Stochastic Systems*, 2013.

Wang, Shusen. A Practical Guide to Randomized Matrix Computations with MATLAB Implementations. *arXiv:1505.07570*, 2015.

White, Martha. Unifying task specification in reinforcement learning. *arXiv:1609.01995*, 2016.

Woodruff, David P. Sketching as a tool for numerical linear algebra. *arXiv:1411.4357*, 2014.

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