Stochastic Variance Reduction Methods for Policy Evaluation

Simon S. Du [1] **Jianshu Chen** [2] **Lihong Li** [2] **Lin Xiao** [2] **Dengyong Zhou** [2]

Abstract

Policy evaluation is concerned with estimating the value function that predicts long-term values of states under a given policy. It is a crucial step in many reinforcement-learning algorithms. In this paper, we focus on policy evaluation with linear function approximation over a *fixed* dataset. We first transform the empirical policy evaluation problem into a (quadratic) convex-concave saddle-point problem, and then present a primal-dual batch gradient method, as well as two stochastic variance reduction methods for solving the problem. These algorithms scale linearly in both sample size and feature dimension. Moreover, they achieve *linear* convergence even when the saddle-point problem has only strong concavity in the dual variables but *no* strong convexity in the primal variables. Numerical experiments on benchmark problems demonstrate the effectiveness of our methods.

1 Introduction

Reinforcement learning (RL) is a powerful learning paradigm for sequential decision making (see, e.g., Bertsekas & Tsitsiklis, 1995; Sutton & Barto, 1998). An RL agent interacts with the environment by repeatedly observing the current state, taking an action according to a certain policy, receiving a reward signal and transitioning to a next state. A policy specifies which action to take given the current state. *Policy evaluation* estimates a value function that predicts expected cumulative reward the agent would receive by following a fixed policy starting at a certain state. In addition to quantifying long-term values of states, which can be of interest on its own, value functions also provide important information for the agent to optimize its policy. For example, *policy-iteration* algorithms iterate between policy-evaluation steps and policy-*improvement* steps, until a (near-)optimal policy is found (Bertsekas & Tsitsiklis, 1995; Lagoudakis & Parr, 2003). Therefore, estimating the value function efficiently and accurately is essential in RL.

There has been substantial work on policy evaluation, with *temporal-difference* (TD) methods being perhaps the most popular. These methods use the Bellman equation to bootstrap the estimation process. Different cost functions are formulated to exploit this idea, leading to different policy evaluation algorithms; see Dann et al. (2014) for a comprehensive survey. In this paper, we study policy evaluation by minimizing the mean squared projected Bellman error (MSPBE) with linear approximation of the value function. We focus on the batch setting where a fixed, finite dataset is given. This fixed-data setting is not only important in itself (Lange et al., 2011), but also an important component in other RL methods such as *experience replay* (Lin, 1992).

The finite-data regime makes it possible to solve policy evaluation more efficiently with recently developed fast optimization methods based on *stochastic variance reduction*, such as SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014). For minimizing strongly convex functions with a finite-sum structure, such methods enjoy the same low computational cost per iteration as the classical stochastic gradient method, but also achieve fast, linear convergence rates (i.e., exponential decay of the optimality gap in the objective). However, they cannot be applied directly to minimize the MSPBE, whose objective does not have the finite-sum structure. In this paper, we overcome this obstacle by transforming the empirical MSPBE problem to an *equivalent* convex-concave saddle-point problem that possesses the desired finite-sum structure.

In the saddle-point problem, we consider the model parameters as the primal variables, which are coupled with the dual variables through a bilinear term. Moreover, without an ℓ_2-regularization on the model parameters, the objective is only strongly concave in the dual variables, but *not* in the primal variables. We propose a primal-dual batch gradient method, as well as two stochastic variance-reduction methods based on SVRG and SAGA, respectively. Surprisingly, we show that when the coupling matrix is full rank, these algorithms achieve linear convergence in both the primal

[1] Machine Learning Department, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA. [2] Microsoft Research, Redmond, Washington 98052, USA.. Correspondence to: Simon S. Du <ssdu@cs.cmu.edu>, Jianshu Chen <jianshuc@microsoft.com>, Lihong Li <lihongli@microsoft.com>, Lin Xiao <lin.xiao@microsoft.com>, Dengyong Zhou <denzho@microsoft.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

and dual spaces, despite the lack of strong convexity of the objective in the primal variables. Our results also extend to *off-policy* learning and TD with *eligibility traces* (Sutton & Barto, 1998; Precup et al., 2001).

We note that Balamurugan & Bach (2016) have extended both SVRG and SAGA to solve convex-concave saddle-point problems with linear-convergence guarantees. The main difference between our results and theirs are

- Linear convergence in Balamurugan & Bach (2016) relies on the assumption that the objective is strongly convex in the primal variables and strongly concave in the dual. Our results show, somewhat surprisingly, that only one of them is necessary if the primal-dual coupling is bilinear and the coupling matrix is full rank. In fact, we are not aware of similar previous results even for the primal-dual batch gradient method, which we show in this paper.
- Even if a strongly convex regularization on the primal variables is introduced to the MSPBE objective, the algorithms in Balamurugan & Bach (2016) cannot be applied efficiently. Their algorithms require that the proximal mappings of the strongly convex and concave regularization functions be computed efficiently. In our saddle-point formulation, the strong concavity of the dual variables comes from a quadratic function defined by the feature covariance matrix, which cannot be inverted efficiently and makes the proximal mapping costly to compute. Instead, our algorithms only use its (stochastic) gradients and hence are much more efficient.

We compare various gradient based algorithms on a Random MDP and Mountain Car data sets. The experiments demonstrate the effectiveness of our proposed methods.

2 Preliminaries

We consider a *Markov Decision Process* (MDP) (Puterman, 2005) described by $(\mathcal{S}, \mathcal{A}, \mathcal{P}_{ss'}^a, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{P}_{ss'}^a$ the transition probability from state s to state s' after taking action a, $\mathcal{R}(s,a)$ the reward received after taking action a in state s, and $\gamma \in [0,1)$ a discount factor. The goal of an agent is to find an action-selection policy π, so that the long-term reward under this policy is maximized. For ease of exposition, we assume $\mathcal{S}$ is finite, but none of our results relies on this assumption.

A key step in many algorithms in RL is to estimate the value function of a given policy π, defined as $V^\pi(s) \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) | s_0 = s, \pi]$. Let V^π denote a vector constructed by stacking the values of $V^\pi(1), \ldots, V^\pi(|\mathcal{S}|)$ on top of each other. Then V^π is the unique fixed point of the *Bellman operator* T^π:

$$V^\pi = T^\pi V^\pi \triangleq R^\pi + \gamma P^\pi V^\pi \,, \tag{1}$$

where R^π is the expected reward vector under policy π, defined elementwise as $R^\pi(s) = \mathbb{E}_{\pi(a|s)} \mathcal{R}(s,a)$; and P^π is the transition matrix induced by the policy applying π, defined entrywise as $P^\pi(s,s') = \mathbb{E}_{\pi(a|s)} \mathcal{P}_{ss'}^a$.

2.1 Mean squared projected Bellman error (MSPBE)

One approach to scale up when the state space size $|\mathcal{S}|$ is large or infinite is to use a linear approximation for V^π. Formally, we use a feature map $\phi : \mathcal{S} \to \mathbb{R}^d$ and approximate the value function by $\widehat{V}^\pi(s) = \phi(s)^T \theta$, where $\theta \in \mathbb{R}^d$ is the model parameter to be estimated. Here, we want to find θ that minimizes the mean squared projected Bellman error, or MSPBE:

$$\text{MSPBE}\,(\theta) \triangleq \frac{1}{2} \| \widehat{V}^\pi - \Pi T^\pi \widehat{V}^\pi \|_\Xi^2, \tag{2}$$

where Ξ is a diagonal matrix with diagonal elements being the stationary distribution over $\mathcal{S}$ induced by the policy π, and Π is the weighted projection matrix onto the linear space spanned by $\phi(1), \ldots, \phi(|\mathcal{S}|)$, that is,

$$\Pi = \Phi(\Phi^T \Xi \Phi)^{-1} \Phi^T \Xi \tag{3}$$

where $\Phi \triangleq [\phi^T(1), \ldots, \phi^T(|\mathcal{S}|)]$ is the matrix obtained by stacking the feature vectors row by row. Substituting (3) and (1) into (2), we obtain (see, e.g., Dann et al., 2014)

$$\text{MSPBE}(\theta) = \frac{1}{2} \| \Phi^T \Xi (\widehat{V}^\pi - T^\pi \widehat{V}^\pi) \|_{(\Phi^T \Xi \Phi)^{-1}}^2 .$$

We can further rewrite the above expression for MSPBE as a standard weighted least-squares problem:

$$\text{MSPBE}(\theta) = \frac{1}{2} \| A\theta - b \|_{C^{-1}}^2 ,$$

with properly defined A, b and C, described as follows. Suppose the MDP under policy π settles at its stationary distribution and generates an infinite transition sequence $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{\infty}$, where s_t is the current state, a_t is the action, r_t is the reward, and s_{t+1} is the next state. Then with the definitions $\phi_t \triangleq \phi(s_t)$ and $\phi'_t \triangleq \phi(s_{t+1})$, we have

$$A = \mathbb{E}[\phi_t(\phi_t - \gamma\phi'_t)^T], \; b = \mathbb{E}[\phi_t r_t], \; C = \mathbb{E}[\phi_t \phi_t^T], \tag{4}$$

where $\mathbb{E}[\cdot]$ are with respect to the stationary distribution. Many TD solutions converge to a minimizer of MSPBE in the limit (Tsitsiklis & Van Roy, 1997; Dann et al., 2014).

2.2 Empirical MSPBE

In practice, quantities in (4) are often unknown, and we only have access to a finite dataset with n transitions $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^n$. By replacing the unknown statistics with their finite-sample estimates, we obtain the Empirical MSPBE, or EM-MSPBE. Specifically, let

$$\widehat{A} \triangleq \frac{1}{n} \sum_{t=1}^n A_t, \quad \widehat{b} \triangleq \frac{1}{n} \sum_{t=1}^n b_t, \quad \widehat{C} \triangleq \frac{1}{n} \sum_{t=1}^n C_t, \tag{5}$$

where for $t = 1, \dots, n$,

$$A_t \triangleq \phi_t(\phi_t - \gamma\phi_t')^T, \quad b_t \triangleq r_t\phi_t, \quad C_t \triangleq \phi_t\phi_t^T. \qquad (6)$$

EM-MSPBE with an *optional* ℓ_2-regularization is given by:

$$\text{EM-MSPBE}\,(\theta) = \frac{1}{2}\|\widehat{A}\theta - \widehat{b}\|^2_{\widehat{C}^{-1}} + \frac{\rho}{2}\|\theta\|^2, \qquad (7)$$

where $\rho \geq 0$ is a regularization factor.

Observe that (7) is a (regularized) weighted least squares problem. Assuming $\widehat{C}$ is invertible, its optimal solution is

$$\theta^\star = (\widehat{A}^\top\widehat{C}^{-1}\widehat{A} + \rho I)^{-1}\widehat{A}^\top\widehat{C}^{-1}\widehat{b}. \qquad (8)$$

Computing $\theta^\star$ directly requires $O(nd^2)$ operations to form the matrices $\widehat{A}$, $\widehat{b}$ and $\widehat{C}$, and then $O(d^3)$ operations to complete the calculation. This method, known as least-squares temporal difference or LSTD (Bradtke & Barto, 1996; Boyan, 2002), can be very expensive when n and d are large. One can also skip forming the matrices explicitly and compute $\theta^\star$ using n recusive rank-one updates (Nedić & Bertsekas, 2003). Since each rank-one update costs $O(d^2)$, the total cost is $O(nd^2)$.

In the sequel, we develop efficient algorithms to minimize EM-MSPBE by using stochastic variance reduction methods, which samples one (ϕ_t, ϕ_t') per update without pre-computing $\widehat{A}$, $\widehat{b}$ and $\widehat{C}$. These algorithms not only maintain a low $O(d)$ per-iteration computation cost, but also attain fast linear convergence rates with a $\log(1/\epsilon)$ dependence on the desired accuracy ϵ.

3 Saddle-Point Formulation of EM-MSPBE

Our algorithms (in Section 5) are based on the stochastic variance reduction techniques developed for minimizing a finite sum of convex functions, more specifically, SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014). They deal with problems of the form

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}, \qquad (9)$$

where each f_i is convex. We immediately notice that the EM-MSPBE in (7) *cannot* be put into such a form, even though the matrices $\widehat{A}$, $\widehat{b}$ and $\widehat{C}$ have the finite-sum structure given in (5). Thus, extending variance reduction techniques to EM-MSPBE minimization is not straightforward.

Nevertheless, we will show that the minimizing the EM-MSPBE is equivalent to solving a convex-concave saddle-point problem which actually possesses the desired finite-sum structure. To proceed, we resort to the machinery of *conjugate functions* (e.g. Rockafellar, 1970, Section 12). For a function $f : \mathbb{R}^d \to \mathbb{R}$, its conjugate function $f^\star : \mathbb{R}^d \to \mathbb{R}$ is defined as $f^\star(y) \triangleq \sup_x(y^Tx - f(x))$. Note that the conjugate function of $\frac{1}{2}\|x\|^2_{\widehat{C}}$ is $\frac{1}{2}\|y\|^2_{\widehat{C}^{-1}}$, i.e.,

$$\frac{1}{2}\|y\|^2_{\widehat{C}^{-1}} = \max_x \left(y^Tx - \frac{1}{2}\|x\|^2_{\widehat{C}} \right).$$

With this relation, we can rewrite EM-MSPBE in (7) as

$$\max_w \left(w^T(\widehat{b} - \widehat{A}\theta) - \frac{1}{2}\|w\|^2_{\widehat{C}} \right) + \frac{\rho}{2}\|\theta\|^2,$$

so that minimizing EM-MSPBE is equivalent to solving

$$\min_{\theta\in\mathbb{R}^d}\max_{w\in\mathbb{R}^d} \left\{ \mathcal{L}(\theta, w) = \frac{1}{n}\sum_{t=1}^{n}\mathcal{L}_t(\theta, w) \right\}, \qquad (10)$$

where the Lagrangian, defined as

$$\mathcal{L}(\theta, w) \triangleq \frac{\rho}{2}\|\theta\|^2 - w^T\widehat{A}\theta - \left(\frac{1}{2}\|w\|^2_{\widehat{C}} - w^T\widehat{b}\right), \qquad (11)$$

may be decomposed using (5), with

$$\mathcal{L}_t(\theta, w) \triangleq \frac{\rho}{2}\|\theta\|^2 - w^TA_t\theta - \left(\frac{1}{2}\|w\|^2_{C_t} - w^Tb_t\right).$$

Therefore, minimizing the EM-MSPBE is equivalent to solving the saddle-point problem (10), which is convex in the primal variable θ and concave in the dual variable w. Moreover, it has a finite-sum structure similar to (9).

Liu et al. (2015) and Valcarcel Macua et al. (2015) independently showed that the GTD2 algorithm (Sutton et al., 2009b) is indeed a *stochastic gradient* method for solving the saddle-point problem (10), although they obtained the saddle-point formulation with different derivations. More recently, Dai et al. (2016) used the conjugate function approach to obtain saddle-point formulations for a more general class of problems and derived primal-dual stochastic gradient algorithms for solving them. However, these algorithms have sublinear convergence rates, which leaves much room to improve when applied to problems with finite datasets. Recently, Lian et al. (2017) developed SVRG methods for a general finite-sum composition optimization that achieve linear convergence rate. Different from our methods, their stochastic gradients are biased and they have worse dependency on the condition numbers (κ^3 and κ^4).

The fast linear convergence of our algorithms presented in Sections 4 and 5 requires the following assumption:

Assumption 1. *$\widehat{A}$ has full rank, $\widehat{C}$ is strictly positive definite, and the feature vector ϕ_t is uniformly bounded.*

Under mild regularity conditions (e.g., Wasserman, 2013, Chapter 5), we have $\widehat{A}$ and $\widehat{C}$ converge in probability to A and C defined in (4), respectively. Thus, if the true statistics A is non-singular and C is positive definite, and we have enough training samples, these assumptions are usually satisfied. They have been widely used in previous works on gradient-based algorithms (e.g., Sutton et al., 2009a;b).

A direct consequence of Assumption 1 is that $\theta^\star$ in (8) is the unique minimizer of the EM-MSPBE in (7), even without any strongly convex regularization on θ (i.e., even if $\rho = 0$). However, if $\rho = 0$, then the Lagrangian $\mathcal{L}(\theta, w)$ is only strongly concave in w, but not strongly convex in θ. In this case, we will show that non-singularity of the coupling matrix $\widehat{A}$ can "pass" an implicit strong convexity on θ, which is exploited by our algorithms to obtain linear convergence in both the primal and dual spaces.

4 A Primal-Dual Batch Gradient Method

Before diving into the stochastic variance reduction algorithms, we first present Algorithm 1, which is a primal-dual *batch* gradient (PDBG) algorithm for solving the saddle-point problem (10). In Step 2, the vector $B(\theta, w)$ is obtained by stacking the primal and negative dual gradients:

$$B\left(\theta, w\right) \triangleq \begin{bmatrix} \nabla_\theta L(\theta, w) \\ -\nabla_w L(\theta, w) \end{bmatrix} = \begin{bmatrix} \rho\theta - \widehat{A}^T w \\ \widehat{A}\theta - \widehat{b} + \widehat{C}w \end{bmatrix}. \qquad (12)$$

Some notation is needed in order to characterize the convergence rate of Algorithm 1. For any symmetric and positive definite matrix S, let $\lambda_{\max}(S)$ and $\lambda_{\min}(S)$ denote its maximum and minimum eigenvalues respectively, and define its condition number to be $\kappa(S) \triangleq \lambda_{\max}(S)/\lambda_{\min}(S)$. We also define L_ρ and μ_ρ for any $\rho \geq 0$:

$$L_\rho \triangleq \lambda_{\max}(\rho I + \widehat{A}^T \widehat{C}^{-1} \widehat{A}), \qquad (13)$$

$$\mu_\rho \triangleq \lambda_{\min}(\rho I + \widehat{A}^T \widehat{C}^{-1} \widehat{A}). \qquad (14)$$

By Assumption 1, we have $L_\rho \geq \mu_\rho > 0$. The following theorem is proved in Appendix B.

Theorem 1. *Suppose Assumption 1 holds and let $(\theta_\star, w_\star)$ be the (unique) solution of (10). If the step sizes are chosen as $\sigma_\theta = \frac{1}{9 L_\rho \kappa(\widehat{C})}$ and $\sigma_w = \frac{8}{9\lambda_{\max}(\widehat{C})}$, then the number of iterations of Algorithm 1 to achieve $\|\theta - \theta_\star\|^2 + \|w - w_\star\|^2 \leq \epsilon^2$ is upper bounded by*

$$O\left(\kappa\left(\rho I + \widehat{A}^T \widehat{C}^{-1} \widehat{A}\right) \cdot \kappa(\widehat{C}) \cdot \log\left(\frac{1}{\epsilon}\right)\right). \qquad (15)$$

We assigned specific values to the step sizes σ_θ and σ_w for clarity. In general, we can use similar step sizes while keeping their ratio roughly constant as $\frac{\sigma_w}{\sigma_\theta} \approx \frac{8 L_\rho}{\lambda_{\min}(\widehat{C})}$; see Appendices A and B for more details. In practice, one can use a parameter search on a small subset of data to find reasonable step sizes. It is an interesting open problem how to automatically select and adjust step sizes.

Note that the linear rate is determined by two parts: (i) the strongly convex regularization parameter ρ, and (ii) the positive definiteness of $\widehat{A}^T \widehat{C}^{-1} \widehat{A}$. The second part could be interpreted as transferring strong concavity in dual variables via the full-rank bi-linear coupling matrix $\widehat{A}$. For this reason, even if the saddle-point problem (10) has only strong concavity in dual variables (when $\rho = 0$), the algorithm still enjoys a linear convergence rate.

Algorithm 1 PDBG for Policy Evaluation

```
Inputs: initial point $(\theta, w)$, step sizes $\sigma_\theta$ and $\sigma_w$, and
    number of epochs $M$.
1: for $i = 1$ to $M$ do
2:     $\begin{bmatrix} \theta \\ w \end{bmatrix} \leftarrow \begin{bmatrix} \theta \\ w \end{bmatrix} - \begin{bmatrix} \sigma_\theta & 0 \\ 0 & \sigma_w \end{bmatrix} B(\theta, w)$
       where $B(\theta, w)$ is computed according to (12).
3: end for
```

Moreover, even if $\rho > 0$, it will be inefficient to solve problem (10) using primal-dual algorithms based on proximal mappings of the strongly convex and concave terms (e.g., Chambolle & Pock, 2011; Balamurugan & Bach, 2016). The reason is that, in (10), the strong concavity of the Lagrangian with respect to the dual lies in the quadratic function $(1/2)\|w\|_{\widehat{C}}$, whose proximal mapping cannot be computed efficiently. In contrast, the PDBG algorithm only needs its gradients.

If we pre-compute and store $\widehat{A}$, $\widehat{b}$ and $\widehat{C}$, which costs $O(nd^2)$ operations, then computing the gradient operator $B(\theta, w)$ in (12) during each iteration of PDBG costs $O(d^2)$ operations. Alternatively, if we do not want to store these $d \times d$ matrices (especially if d is large), then we can compute $B(\theta, w)$ as finite sums on the fly. More specifically, $B(\theta, w) = \frac{1}{n}\sum_{t=1}^n B_t(\theta, w)$, where for each $t = 1, \ldots, n$,

$$B_t(\theta, w) = \begin{bmatrix} \rho\theta - A_t w \\ A_t\theta - b_t + C_t w \end{bmatrix}. \qquad (16)$$

Since A_t, b_t and C_t are all rank-one matrices, as given in (6), computing each $B_t(\theta, w)$ only requires $O(d)$ operations. Therefore, computing $B(\theta, w)$ costs $O(nd)$ operations as it averages $B_t(\theta, w)$ over n samples.

5 Stochastic Variance Reduction Methods

If we replace $B(\theta, w)$ in Algorithm 1 (line 2) by the stochastic gradient $B_t(\theta, w)$ in (16), then we recover the GTD2 algorithm of Sutton et al. (2009b), applied to a fixed dataset, possibly with *multiple passes*. It has a low per-iteration cost but a slow, *sublinear* convergence rate. In this section, we provide two stochastic variance reduction methods and show they achieve fast linear convergence.

5.1 SVRG for policy evaluation

Algorithm 2 is adapted from the stochastic variance reduction gradient (SVRG) method (Johnson & Zhang, 2013). It uses two layers of loops and maintains two sets of parameters $(\tilde{\theta}, \tilde{w})$ and (θ, w). In the outer loop, the algorithm computes a full gradient $B(\tilde{\theta}, \tilde{w})$ using $(\tilde{\theta}, \tilde{w})$, which takes

Algorithm 2 SVRG for Policy Evaluation

```
Inputs: initial point $(\theta, w)$, step sizes $\{\sigma_\theta, \sigma_w\}$, number of
    outer iterations $M$, and number of inner iterations $N$.
for $m = 1$ to $M$ do
    Initialize $(\tilde{\theta}, \tilde{w}) = (\theta, w)$ and compute $B(\tilde{\theta}, \tilde{w})$.
    for $j = 1$ to $N$ do
        Sample an index $t_j$ from $\{1, \cdots, n\}$ and do
        Compute $B_{t_j}(\theta, w)$ and $B_{t_j}(\tilde{\theta}, \tilde{w})$.
        $\begin{bmatrix}\theta \\ w\end{bmatrix} \leftarrow \begin{bmatrix}\theta \\ w\end{bmatrix} - \begin{bmatrix}\sigma_\theta & 0 \\ 0 & \sigma_w\end{bmatrix} B_{t_j}(\theta, w, \tilde{\theta}, \tilde{w})$
        where $B_{t_j}(\theta, w, \tilde{\theta}, \tilde{w})$ is given in (17).
    end for
end for
```

Algorithm 3 SAGA for Policy Evaluation

```
Inputs: initial point $(\theta, w)$, step sizes $\sigma_\theta$ and $\sigma_w$, and
    number of iterations $M$.
Compute each $g_t = B_t(\theta, w)$ for $t = 1, \ldots, n$.
Compute $B = B(\theta, w) = \frac{1}{n}\sum_{t=1}^n g_t$.
for $m = 1$ to $M$ do
    Sample an index $t_m$ from $\{1, \cdots, n\}$.
    Compute $h_{t_m} = B_{t_m}(\theta, w)$.
    $\begin{bmatrix}\theta \\ w\end{bmatrix} \leftarrow \begin{bmatrix}\theta \\ w\end{bmatrix} - \begin{bmatrix}\sigma_\theta & 0 \\ 0 & \sigma_w\end{bmatrix} (B + h_{t_m} - g_{t_m})$.
    $B \leftarrow B + \frac{1}{n}(h_{t_m} - g_{t_m})$
    $g_{t_m} \leftarrow h_{t_m}$.
end for
```

$O(nd)$ operations. Afterwards, the algorithm executes the inner loop, which randomly samples an index t_j and updates (θ, w) using variance-reduced stochastic gradient:

$$B_{t_j}(\theta, w, \tilde{\theta}, \tilde{w}) = B_{t_j}(\theta, w) + B(\tilde{\theta}, \tilde{w}) - B_{t_j}(\tilde{\theta}, \tilde{w}). \quad (17)$$

Here, $B_{t_j}(\theta, w)$ contains the stochastic gradients at (θ, w) computed using the random sample with index t_j, and $B(\tilde{\theta}, \tilde{w}) - B_{t_j}(\tilde{\theta}, \tilde{w})$ is a term used to reduce the variance in $B_{t_j}(\theta, w)$ while keeping $B_{t_j}(\theta, w, \tilde{\theta}, \tilde{w})$ an unbiased estimate of $B(\theta, w)$.

Since $B(\tilde{\theta}, \tilde{w})$ is computed once during each iteration of the outer loop with cost $O(nd)$ (as explained at the end of Section 4), and each of the N iterations of the inner loop cost $O(d)$ operations, the total computational cost of for each outer loop is $O(nd+Nd)$. We will present the overall complexity analysis of Algorithm 2 in Section 5.3.

5.2 SAGA for policy evaluation

The second stochastic variance reduction method for policy evaluation is adapted from SAGA (Defazio et al., 2014); see Algorithm 3. It uses a single loop, and maintains a single set of parameters (θ, w). Algorithm 3 starts by first computing each component gradients $g_t = B_t(\theta, w)$ at the initial point, and also form their average $B = \sum_t^n g_t$. At each iteration, the algorithm randomly picks an index $t_m \in \{1, \ldots, n\}$ and computes the stochastic gradient $h_{t_m} = B_{t_m}(\theta, w)$. Then, it updates (θ, w) using a variance reduced stochastic gradient: $B + h_{t_m} - g_{t_m}$, where g_{t_m} is the previously computed stochastic gradient using the t_m-th sample (associated with certain past values of θ and w). Afterwards, it updates the batch gradient estimate B as $B + \frac{1}{n}(h_{t_m} - g_{t_m})$ and replaces g_{t_m} with h_{t_m}.

As Algorithm 3 proceeds, different vectors g_t are computed using different values of θ and w (depending on when the index t was sampled). So in general we need to store all vectors g_t, for $t = 1, \ldots, n$, to facilitate individual updates, which will cost additional $O(nd)$ storage. However, by exploiting the rank-one structure in (6), we only need to store three scalars $(\phi_t - \gamma\phi_t')^T\theta$, $(\phi_t - \gamma\phi_t')^T w$, and $\phi_t^T w$, and form g_{t_m} on the fly using $O(d)$ computation. Overall, each iteration of SAGA costs $O(d)$ operations.

5.3 Theoretical analyses of SVRG and SAGA

In order to study the convergence properties of SVRG and SAGA for policy evaluation, we introduce a smoothness parameter L_G based on the stochastic gradients $B_t(\theta, w)$. Let $\beta = \sigma_w/\sigma_\theta$ be the ratio between the primal and dual step-sizes, and define a pair of weighted Euclidean norms

$$\Omega(\theta, w) \triangleq (\|\theta\|^2 + \beta^{-1}\|w\|^2)^{1/2},$$
$$\Omega^*(\theta, w) \triangleq (\|\theta\|^2 + \beta\|w\|^2)^{1/2}.$$

Note that $\Omega(\cdot, \cdot)$ upper bounds the error in optimizing θ: $\Omega(\theta - \theta_\star, w - w_\star) \geq \|\theta - \theta_\star\|$. Therefore, any bound on $\Omega(\theta - \theta_\star, w - w_\star)$ applies automatically to $\|\theta - \theta_\star\|$.

Next, we define the parameter L_G through its square:

$$L_G^2 \triangleq \sup_{\theta_1, w_1, \theta_2, w_2} \frac{\frac{1}{n}\sum_{t=1}^n \Omega^*\big(B_t(\theta_1, w_1) - B_t(\theta_2, w_2)\big)^2}{\Omega(\theta_1 - \theta_2, w_1 - w_2)^2}.$$

This definition is similar to the smoothness constant $\bar{L}$ used in Balamurugan & Bach (2016) except that we used the step-size ratio β rather than the strong convexity and concavity parameters of the Lagrangian to define Ω and Ω^*.[1] Substituting the definition of $B_t(\theta, w)$ in (16), we have

$$L_G^2 = \left\|\frac{1}{n}\sum_{t=1}^n G_t^T G_t\right\|, \text{ where } G_t \triangleq \begin{bmatrix}\rho I & -\sqrt{\beta}A_t^T \\ \sqrt{\beta}A_t & \beta C_t\end{bmatrix}. \quad (18)$$

With the above definitions, we characterize the convergence of $\Omega(\theta_m - \theta_\star, w_m - w_\star)$, where $(\theta_\star, w_\star)$ is the solution of (10), and (θ_m, w_m) is the output of the algorithms

[1] Since our saddle-point problem is not necessarily strongly convex in θ (when $\rho = 0$), we could not define Ω and Ω^* in the same way as Balamurugan & Bach (2016).

after the m-th iteration. For SVRG, it is the m-th *outer* iteration in Algorithm 2. The following two theorems are proved in Appendices C and D, respectively.

Theorem 2 (Convergence rate of SVRG)**.** *Suppose Assumption 1 holds. If we choose* $\sigma_\theta = \frac{\mu_\rho}{48\kappa(\widehat{C})L_G^2}$, $\sigma_w = \frac{8L_\rho}{\lambda_{\min}(\widehat{C})}\sigma_\theta$, $N = \frac{51\kappa^2(\widehat{C})L_G^2}{\mu_\rho^2}$, *where* L_ρ *and* μ_ρ *are defined in* (13) *and* (14)*, then*

$$\mathbb{E}\big[\Omega(\theta_m-\theta_\star, w_m-w_\star)^2\big] \le \left(\frac{4}{5}\right)^m \Omega(\theta_0-\theta_\star, w_0-w_\star)^2.$$

The overall computational cost for reaching $\mathbb{E}\big[\Omega(\theta_m - \theta_\star, w_m - w_\star)\big] \le \epsilon$ *is upper bounded by*

$$O\left(\left(n + \frac{\kappa(\widehat{C})L_G^2}{\lambda_{\min}^2(\rho I + \widehat{A}^T\widehat{C}^{-1}\widehat{A})}\right)d\,\log\left(\frac{1}{\epsilon}\right)\right). \tag{19}$$

Theorem 3 (Convergence rate of SAGA)**.** *Suppose Assumption 1 holds. If we choose* $\sigma_\theta = \frac{\mu_\rho}{3\left(8\kappa^2(\widehat{C})L_G^2+n\mu_\rho^2\right)}$ *and* $\sigma_w = \frac{8L_\rho}{\lambda_{\min}(\widehat{C})}\sigma_\theta$ *in Algorithm 3, then*

$$\mathbb{E}\big[\Omega(\theta_m-\theta_\star, w_m-w_\star)^2\big] \le 2(1-\rho)^m\Omega(\theta_0-\theta_\star, w_0-w_\star)^2,$$

where $\rho \ge \frac{\mu_\rho^2}{9\left(8\kappa^2(\widehat{C})L_G^2+n\mu_\rho^2\right)}$. *The total cost to achieve* $\mathbb{E}\big[\Omega(\theta_m - \theta_\star, w_m - w_\star)\big] \le \epsilon$ *has the same bound in* (19)*.*

Similar to our PDBG results in (15), both the SVRG and SAGA algorithms for policy evaluation enjoy linear convergence even if there is no strong convexity in the saddle-point problem (10) (i.e., when $\rho = 0$). This is mainly due to the positive definiteness of $\widehat{A}^T\widehat{C}^{-1}\widehat{A}$ when $\widehat{C}$ is positive-definite and $\widehat{A}$ is full-rank. In contrast, the linear convergence of SVRG and SAGA in Balamurugan & Bach (2016) requires the Lagrangian to be both strongly convex in θ and strongly concave in w.

Moreover, in the policy evaluation problem, the strong concavity with respect to the dual variable w comes from a weighted quadratic norm $(1/2)\|w\|_{\widehat{C}}$, which does not admit an efficient proximal mapping as required by the proximal versions of SVRG and SAGA in Balamurugan & Bach (2016). Our algorithms only require computing the stochastic gradients of this function, which is easy to do due to its finite sum structure.

Balamurugan & Bach (2016) also proposed accelerated variants of SVRG and SAGA using the "catalyst" framework of Lin et al. (2015). Such extensions can be done similarly for the three algorithms presented in this paper, and we omit the details due to space limit.

6 Comparison of Different Algorithms

This section compares the computation complexities of several representative policy-evaluation algorithms that minimize EM-MSPBE, as summarized in Table 1.

Table 1. Complexity of different policy evaluation algorithms. In the table, d is feature dimension, n is dataset size, $\kappa \triangleq \kappa(\rho I + \widehat{A}^T\widehat{C}^{-1}\widehat{A})$; $\kappa_G \triangleq L_G/\lambda_{\min}(\rho I + \widehat{A}^T\widehat{C}^{-1}\widehat{A})$; and κ' is a condition number related to GTD2.

Algorithm	**Total Complexity**
SVRG / SAGA	$O\left(nd\cdot\left(1+\frac{\kappa(\widehat{C})\kappa_G^2}{n}\right)\cdot\log\left(1/\epsilon\right)\right)$
GTD2	$O\left(d\cdot\kappa'/\epsilon\right)$
PDBG-(I)	$O\left(nd\cdot\kappa(\widehat{C})\kappa\cdot\log(1/\epsilon)\right)$
PDBG-(II)	$O\left(nd^2 + d^2\kappa(\widehat{C})\kappa\cdot\log(1/\epsilon)\right)$
LSTD	$O\big(nd^2\big)$ or $O\big(nd^2+d^3\big)$

The upper part of the table lists algorithms whose complexity is linear in feature dimension d, including the two new algorithms presented in the previous section. We can also apply GTD2 to a finite dataset with samples drawn uniformly at random with replacement. It costs $O(d)$ per iteration, but has a sublinear convergence rate regarding ϵ. In practice, people may choose $\epsilon = \Omega(1/n)$ for generalization reasons (see, e.g., Lazaric et al. (2010)), leading to an $O(\kappa' nd)$ overall complexity for GTD2, where κ' is a condition number related to the algorithm. However, as verified by our experiments, the bounds in the table show that our SVRG/SAGA-based algorithms are much faster as their effective condition numbers vanish when n becomes large. TDC has a similar complexity to GTD2.

In the table, we list two different implementations of PDBG. PDBG-(I) computes the gradients by averaging the stochastic gradients over the entire dataset at each iteration, which costs $O(nd)$ operations; see discussions at the end of Section 4. PDBG-(II) first pre-computes the matrices $\widehat{A}$, $\widehat{b}$ and $\widehat{C}$ using $O(nd^2)$ operations, then computes the batch gradient at each iteration with $O(d^2)$ operations. If d is very large (e.g., when $d \gg n$), then PDBG-(I) would have an advantage over PDBG-(II). The lower part of the table also includes LSTD, which has $O(nd^2)$ complexity if rank-one updates are used.

SVRG and SAGA are more efficient than the other algorithms, when either d or n is very large. In particular, they have a lower complexity than LSTD when $d > (1 + \frac{\kappa(\widehat{C})\kappa_G^2}{n})\log\left(\frac{1}{\epsilon}\right)$. This condition is easy to satisfy, when n is very large. On the other hand, SVRG and SAGA algorithms are more efficient than PDBG-(I) if n is large, say $n > \kappa(\widehat{C})\kappa_G^2/\big(\kappa(\widehat{C})\kappa - 1\big)$, where κ and κ_G are described in the caption of Table 1.

There are other algorithms whose complexity scales linearly with n and d, including iLSTD (Geramifard et al., 2007), and TDC (Sutton et al., 2009b), fLSTD-SA (Prashanth et al., 2014), and the more recent algorithms of Wang et al. (2016) and Dai et al. (2016). However, their convergence is slow: the number of iterations required to reach a desired accuracy ϵ grows as $1/\epsilon$ or worse. The CTD algorithm (Korda & Prashanth, 2015) uses a similar idea as SVRG to reduce variance in TD updates. This algorithm is shown to have a similar linear convergence rate in an *online* setting where the data stream is generated by a Markov process with *finite* states and *exponential* mixing. The method solves for a fixed-point solution by stochastic approximation. As a result, they can be non-convergent in off-policy learning, while our algorithms remain stable (c.f., Section 7.1).

7 Extensions

It is possible to extend our approach to accelerate optimization of other objectives such as MSBE and NEU (Dann et al., 2014). In this section, we briefly describe two extensions of the algorithms developed earlier.

7.1 Off-policy learning

In some cases, we may want to estimate the value function of a policy π from a set of data $\mathcal{D}$ generated by a different "behavior" policy π_b. This is called *off-policy learning* (Sutton & Barto, 1998, Chapter 8).

In the off-policy case, samples are generated from the distribution induced by the behavior policy π_b, not the the target policy π. While such a mismatch often causes stochastic-approximation-based methods to diverge (Tsitsiklis & Van Roy, 1997), our gradient-based algorithms remain convergent with the same (fast) convergence rate.

Consider the RL framework outlined in Section 2. For each state-action pair (s_t, a_t) such that $\pi_b(a_t|s_t) > 0$, we define the importance ratio, $\rho_t \triangleq \pi(a_t|s_t)/\pi_b(a_t|s_t)$. The EM-MSPBE for off-policy learning has the same expression as in (7) except that A_t, b_t and C_t are modified by the weight factor ρ_t, as listed in Table 2; see also Liu et al. (2015, Eqn 6) for a related discussion.) Algorithms 1–3 remain the same for the off-policy case after A_t, b_t and C_t are modified correspondingly.

7.2 Learning with eligibility traces

Eligibility traces are a useful technique to trade off bias and variance in TD learning (Singh & Sutton, 1996; Kearns & Singh, 2000). When they are used, we can pre-compute z_t in Table 2 before running our new algorithms. Note that EM-MSPBE with eligibility traces has the same form of (7), with A_t, b_t and C_t defined differently according to the last row of Table 2. At the m-th step of the learning process, the algorithm randomly samples $z_{t_m}, \phi_{t_m}, \phi'_{t_m}$ and r_{t_m} from the fixed dataset and computes the corresponding stochastic gradients, where the index t_m is uniformly distributed over $\{1, \ldots, n\}$ and are independent for different values of m. Algorithms 1–3 immediately work for this case, enjoying a similar linear convergence rate and a computation complexity linear in n and d. We need additional $O(nd)$ operations to pre-compute z_t recursively and an additional $O(nd)$ storage for z_t. However, it does not change the order of the total complexity for SVRG/SAGA.

Table 2. Expressions of A_t, b_t and C_t for different cases of policy evaluation. Here, $\rho_t \triangleq \pi(a_t|s_t)/\pi_b(a_t|s_t)$; and $z_t \triangleq \sum_{i=1}^{t}(\lambda\gamma)^{t-i}\phi_i$, where $\lambda \geq 0$ is a given parameter.

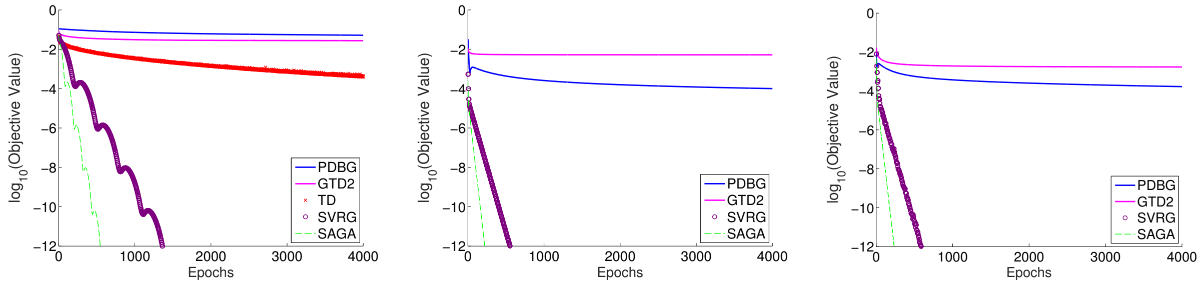
	A_t	b_t	C_t
On-policy	$\phi_t(\phi_t - \gamma\phi'_t)^\top$	$r_t\phi_t$	$\phi_t\phi_t^\top$
Off-policy	$\rho_t\phi_t(\phi_t - \gamma\phi'_t)^\top$	$\rho_t r_t\phi_t$	$\phi_t\phi_t^\top$
Eligibility trace	$z_t(\phi_t - \gamma\phi'_t)^\top$	$r_t z_t$	$\phi_t\phi_t^\top$

8 Experiments

In this section, we compare the following algorithms on two benchmark problems: (i) **PDBG** (Algorithm 1); (ii) **GTD2** with samples drawn randomly with replacement from a dataset; (iii) **TD**: the fLSTD-SA algorithm of Prashanth et al. (2014); (iv) **SVRG** (Algorithm 2); and (v) **SAGA** (Algorithm 3). Note that when $\rho > 0$, the TD solution and EM-MSPBE minimizer differ, so we do not include **TD**. For step size tuning, σ_θ is chosen from $\left\{10^{-1}, 10^{-2}, \ldots, 10^{-6}\right\} \frac{1}{L_\rho \kappa(\widehat{C})}$ and σ_w is chosen from $\left\{1, 10^{-1}, 10^{-2}\right\} \frac{1}{\lambda_{\max}(\widehat{C})}$. We only report the results of each algorithm which correspond to the best-tuned step sizes; for SVRG we choose $N = 2n$.

In the first task, we consider a randomly generated MDP with 400 states and 10 actions (Dann et al., 2014). The transition probabilities are defined as $P\left(s'|a, s\right) \propto p^a_{ss'} + 10^{-5}$, where $p^a_{ss'} \sim U[0, 1]$. The data-generating policy and start distribution were generated in a similar way. Each state is represented by a 201-dimensional feature vector, where 200 of the features were sampled from a uniform distribution, and the last feature was constant one. We chose $\gamma = 0.95$. Fig. 1 shows the performance of various algorithms for $n = 20000$. First, notice that the stochastic variance methods converge much faster than others. In fact, our proposed methods achieve linear convergence. Second, as we increase ρ, the performances of PDBG, SVRG and SAGA improve significantly due to better conditioning, as predicted by our theoretical results.

Next, we test these algorithms on Mountain Car (Sutton & Barto, 1998, Chapter 8). To collect the dataset, we first ran Sarsa with $d = 300$ CMAC features to obtain a good policy. Then, we ran this policy to collect trajectories that comprise the dataset. Figs. 2 and 3 show our proposed stochastic variance reduction methods dominate other first-order methods. Moreover, with better conditioning (through a larger ρ), PDBG, SVRG and SAGA achieve faster convergence rate. Finally, as we increase sample size n, SVRG and SAGA converge faster. This simulation verifies our

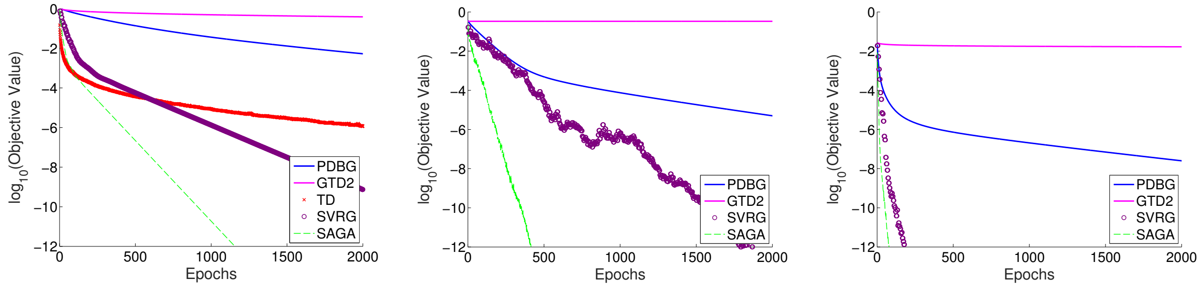


(a) $\rho = 0$

(b) $\rho = \sqrt{\lambda_{\max}(\widehat{A}^\top \widehat{C}^{-1} \widehat{A})}$

(c) $\rho = \lambda_{\max}\left(\widehat{A}^\top \widehat{C}^{-1} \widehat{A}\right)$

Figure 1. Random MDP with $s = 400$, $d = 200$, and $n = 20000$.

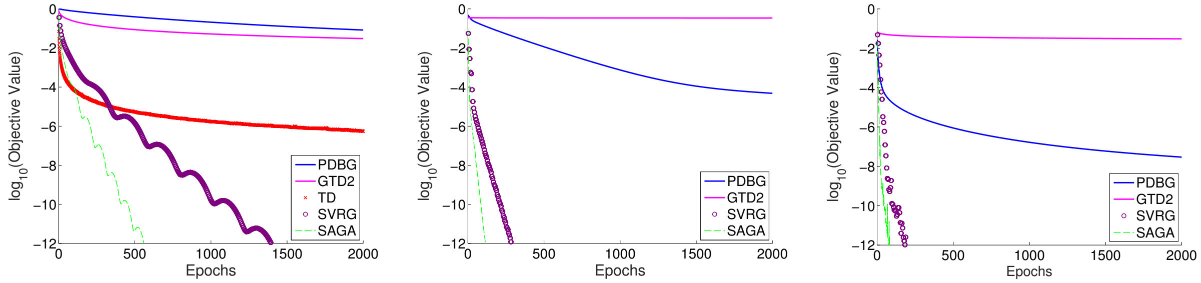


(a) $\rho = 0$

(b) $\rho = 0.01\lambda_{\max}(\widehat{A}^\top \widehat{C}^{-1} \widehat{A})$

(c) $\rho = \lambda_{\max}(\widehat{A}^\top \widehat{C}^{-1} \widehat{A})$

Figure 2. Mountain Car Data Set with $d = 300$ and $n = 5000$.



(a) $\rho = 0$

(b) $\rho = 0.01\lambda_{\max}(\widehat{A}^\top \widehat{C}^{-1} \widehat{A})$

(c) $\rho = \lambda_{\max}(\widehat{A}^\top \widehat{C}^{-1} \widehat{A})$

Figure 3. Mountain Car Data Set with $d = 300$ and $n = 20000$.

theoretical finding in Table 1 that SVRG/SAGA need fewer epochs for large n.

9 Conclusions

In this paper, we reformulated the EM-MSPBE minimization problem in policy evaluation into an empirical saddle-point problem, and developed and analyzed a batch gradient method and two first-order stochastic variance reduction methods to solve the problem. An important result we obtained is that even when the reformulated saddle-point problem lacks strong convexity in primal variables and has only strong concavity in dual variables, the proposed algorithms are still able to achieve a linear convergence rate. We are not aware of any similar results for primal-dual batch gradient methods or stochastic variance reduction methods. Furthermore, we showed that when both the feature dimension d and the number of samples n are large, the developed stochastic variance reduction methods are more efficient than any other gradient-based methods which are convergent in off-policy settings.

This work leads to several interesting directions for research. First, we believe it is important to extend the stochastic variance reduction methods to nonlinear approximation paradigms (Bhatnagar et al., 2009), especially with deep neural networks. Moreover, it remains an important open problem how to apply stochastic variance reduction techniques to policy optimization.

References

Balamurugan, P and Bach, Francis. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems 29*, pp. 1416–1424, 2016.

Bertsekas, Dimitri P and Tsitsiklis, John N. Neuro-dynamic programming: An overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pp. 560–564. IEEE, 1995.

Bhatnagar, Shalabh, Precup, Doina, Silver, David, Sutton, Richard S, Maei, Hamid R, and Szepesvári, Csaba. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pp. 1204–1212, 2009.

Boyan, Justin A. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

Bradtke, Steven J and Barto, Andrew G. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

Chambolle, Antonin and Pock, Thomas. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

Dai, Bo, He, Niao, Pan, Yunpeng, Boots, Byron, and Song, Le. Learning from conditional distributions via dual embeddings. arXiv:1607.04579, 2016.

Dann, Christoph, Neumann, Gerhard, and Peters, Jan. Policy evaluation with temporal differences: a survey and comparison. *Journal of Machine Learning Research*, 15 (1):809–883, 2014.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Geramifard, Alborz, Bowling, Michael H., Zinkevich, Martin, and Sutton, Richard S. iLSTD: Eligibility traces and convergence analysis. In *Advances in Neural Information Processing Systems 19*, pp. 441–448, 2007.

Gohberg, Israel, Lancaster, Peter, and Rodman, Leiba. *Indefinite linear algebra and applications*. Springer Science & Business Media, 2006.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Kearns, Michael J. and Singh, Satinder P. "Bias-variance" error bounds for temporal difference updates. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (COLT-00)*, pp. 142–147, 2000.

Korda, Nathaniel and Prashanth, L.A. On TD(0) with function approximation: Concentration bounds and a centered variant with exponential convergence. In *Proceedings of the Thirty-Second International Conference on Machine Learning (ICML-15)*, pp. 626–634, 2015.

Lagoudakis, Michail G and Parr, Ronald. Least-squares policy iteration. *Journal of Machine Learning Research*, 4(Dec):1107–1149, 2003.

Lange, Sascha, Gabel, Thomas, and Riedmiller, Martin. Batch reinforcement learning. In Wiering, Marco and van Otterlo, Martijn (eds.), *Reinforcement Learning: State of the Art*, pp. 45–73. Springer Verlag, 2011.

Lazaric, Alessandro, Ghavamzadeh, Mohammad, and Munos, Rémi. Finite-sample analysis of LSTD. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pp. 615–622, 2010.

Lian, Xiangru, Wang, Mengdi, and Liu, Ji. Finite-sum composition optimization via variance reduced gradient descent. In *Proceedings of Artificial Intelligence and Statistics Conference (AISTATS)*, pp. 1159–1167, 2017.

Liesen, Jörg and Parlett, Beresford N. On nonsymmetric saddle point matrices that allow conjugate gradient iterations. *Numerische Mathematik*, 108(4):605–624, 2008.

Lin, Hongzhou, Mairal, Julien, and Harchaoui, Zaid. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS) 28*, pp. 3384–3392, 2015.

Lin, Long-Ji. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321, 1992.

Liu, Bo, Liu, Ji, Ghavamzadeh, Mohammad, Mahadevan, Sridhar, and Petrik, Marek. Finite-sample analysis of proximal gradient TD algorithms. In *Proc. The 31st Conf. Uncertainty in Artificial Intelligence, Amsterdam, Netherlands*, 2015.

Nedić, A. and Bertsekas, Dimitri P. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamics Systems: Theory and Applications*, 13(1):79–110, 2003.

Prashanth, LA, Korda, Nathaniel, and Munos, Rémi. Fast LSTD using stochastic approximation: Finite time analysis and application to traffic control. In *Joint European*

Conference on Machine Learning and Knowledge Discovery in Databases, pp. 66–81. Springer, 2014.

Precup, Doina, Sutton, Richard S., and Dasgupta, Sanjoy. Off-policy temporal-difference learning with funtion approximation. In *Proceedings of the Eighteenth Conference on Machine Learning (ICML-01)*, pp. 417–424, 2001.

Puterman, Martin L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.

Rockafellar, R. Tyrrell. *Convex Analysis*. Princeton University Press, 1970.

Shen, Shu-Qian, Huang, Ting-Zhu, and Cheng, Guang-Hui. A condition for the nonsymmetric saddle point matrix being diagonalizable and having real and positive eigenvalues. *Journal of Computational and Applied Mathematics*, 220(1):8–12, 2008.

Singh, Satinder P. and Sutton, Richard S. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1–3):123–158, 1996.

Sutton, Richard S and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Sutton, Richard S, Maei, Hamid R, and Szepesvári, Csaba. A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in neural information processing systems*, pp. 1609–1616, 2009a.

Sutton, Richard S, Maei, Hamid Reza, Precup, Doina, Bhatnagar, Shalabh, Silver, David, Szepesvári, Csaba, and Wiewiora, Eric. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000. ACM, 2009b.

Tsitsiklis, John N. and Van Roy, Benjamin. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42: 674–690, 1997.

Valcarcel Macua, Sergio, Chen, Jianshu, Zazo, Santiago, and Sayed, Ali H. Distributed policy evaluation under multiple behavior strategies. *Automatic Control, IEEE Transactions on*, 60(5):1260–1274, 2015.

Wang, Mengdi, Liu, Ji, and Fang, Ethan. Accelerating stochastic composition optimization. In *Advances in Neural Information Processing Systems (NIPS) 29*, pp. 1714–1722, 2016.

Wasserman, Larry. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, 2013.