SVRG for Policy Evaluation with Fewer Gradient Evaluations

Zilun Peng[1*], **Ahmed Touati**[2†*], **Pascal Vincent**[2,3] and **Doina Precup**[1]
[1]Mila, McGill University
[2]Mila, Université de Montréal
[3]Facebook AI Research
zilun.peng@mail.mcgill.ca {ahmed.touati, pascal.vincent}@umontreal.ca, dprecup@cs.mcgill.ca

Abstract

Stochastic variance-reduced gradient (SVRG) is an optimization method originally designed for tackling machine learning problems with a finite sum structure. SVRG was later shown to work for policy evaluation, a problem in reinforcement learning in which one aims to estimate the value function of a given policy. SVRG makes use of gradient estimates at two scales. At the slower scale, SVRG computes a full gradient over the whole dataset, which could lead to prohibitive computation costs. In this work, we show that two variants of SVRG for policy evaluation could significantly diminish the number of gradient calculations while preserving a linear convergence speed. More importantly, our theoretical result implies that one does not need to use the entire dataset in every epoch of SVRG when it is applied to policy evaluation with linear function approximation. Our experiments demonstrate large computational savings provided by the proposed methods.

1 Introduction

Policy evaluation is an important problem in reinforcement learning (RL) whose goal is to estimate the value function, a function which measures the long-term expected return at any given state. Theoretically well-studied linear methods for policy evaluation fall into two categories: least square approaches [Bradtke and Barto, 1996; Boyan, 2002] and gradient based approaches [Baird, 1995; Sutton *et al.*, 2008; Sutton *et al.*, 2009; Liu *et al.*, 2015]. In large data settings, least square approaches are not efficient because they compute matrix inverses, and gradient based approaches are preferable.

One of objective functions used by gradient-based methods for policy evaluation is Mean Squared Projected Bellman Error (MSPBE). Existing methods such as TD [Sutton, 1988], GTD2 and TDC [Sutton *et al.*, 2009] have sublinear convergence rates when solving this objective as shown in [Bhandari *et al.*, 2018; Touati *et al.*, 2018; Dalal *et al.*, 2019]. Du et al. [2017] applied SVRG [Johnson and Zhang, 2013] to solve MSPBE, leading to a linear convergent method. An important and computationally heavy step of SVRG is to compute a full gradient at the beginning of every epoch.

In this paper, we address the computational bottleneck of SVRG for policy evaluation, by extending batching SVRG [Harikandeh *et al.*, 2015] and SCSG [Lei and Jordan, 2017] to solve MSPBE. Both methods were originally designed for convex minimization problems. These are significantly different from our objective, a convex-concave saddle-point problem without strong convexity in the primal variable[1]. The convergence analysis in [Harikandeh *et al.*, 2015; Lei and Jordan, 2017] does not apply in our settings and we establish new convergence results of batching SVRG and SCSG for solving MSPBE. In this work, we make the following key contributions:

1. We show that both batching SVRG and SCSG achieve linear convergence rates for policy evaluation while yielding considerable savings in the number of gradient computations. To the best of our knowledge, this is the first result for batching SVRG and SCSG in the saddle-point setting.
2. While our analysis of SCSG builds on the ideas of [Lei and Jordan, 2017], our proofs end up quite different and also a lot simpler because we exploit the structure of our problem.
3. Our experimental results demonstrate that batching SVRG and SCSG are efficient in large data settings.

2 Background

In RL, a Markov Decision Process (MDP) is typically used to model the interaction between an agent and its environment. An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of actions,the transition probability function $P : \mathcal{S} \times \mathcal{A} \to (\mathcal{S} \to [0, 1])$ maps state-action pairs to distributions over next states. r denotes the reward function: $(\mathcal{S}, \mathcal{A}) \to \mathbb{R}$, which returns the immediate reward that an agent will receive after performing an action a at state s and $\gamma \in (0, 1)$ is the discount factor used to discount rewards received farther in the future. For simplicity, we will assume $\mathcal{S}$ and $\mathcal{A}$ are finite.

*Equal contribution. † Work done while at Facebook AI research.

[1]A general convex-concave saddle-point problem can be solved with a linear convergence rate [Balamurugan and Bach, 2016], but the method requires strong convexity in the primal variable and the access to a proximal operator.

A policy $\pi : \mathcal{S} \to (\mathcal{A} \to [0, 1])$ is a mapping from states to distributions over actions. The value function for policy π, denoted $V^\pi : \mathcal{S} \to \mathbb{R}$, represents the expected sum of discounted rewards along the trajectories induced by the policy in the MDP: $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi]$. V^π can be obtained as the fixed point of the Bellman operator over the action-value function $\mathcal{T}^\pi V = r^\pi + \gamma P^\pi V$ where r^π is the expected immediate reward and P^π is defined as $P^\pi(s'|s) \triangleq \sum_{a \in \mathcal{A}} \pi(a \mid s) P(s' \mid s, a)$.

In this paper, we are concerned with the policy evaluation problem [Sutton and Barto, 1998] i.e. estimation of V^π for a given policy π. In order to obtain generalization between different states, V^π should be represented in a functional form. In this paper, we focus on linear function approximation of the form: $V(s) \triangleq \theta^\top \phi(s)$ where $\theta \in \mathbb{R}^d$ is a weight vector and $\phi : \mathcal{S} \to \mathbb{R}^d$ is a feature map from states to a given d-dimensional feature space.

3 Objective Functions

We assume that the Markov chain induced by the policy π is ergodic and admits a unique stationary distribution, denoted by d^π, over states. We write D^π for the diagonal matrix whose diagonal entries are $(d^\pi(s))_{s \in \mathcal{S}}$.

If $\Phi = (\phi(s))_{s \in \mathcal{S}} \in \mathbb{R}^{|\mathcal{S}| \times d}$ denotes the matrix obtained by stacking the state feature vectors row by row, then it is known [Bertsekas, 2011] that $\Phi\theta^\star$ is the fixed point of the projected Bellman operator :

$$\Phi\theta^* = \Pi^\pi \mathcal{T}^\pi(\Phi\theta^*) \ , \tag{1}$$

where $\Pi^\pi = \Phi(\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi$ is the orthogonal projection onto the space $S = \{\Phi\theta \mid \theta \in \mathbb{R}^d\}$ with respect to the weighted Euclidean norm $||.||_{D^\pi}$. Rather than computing a sequence of iterates given by the projected Bellman operator, another approach for finding θ^* is to directly minimize [Sutton *et al.*, 2009; Liu *et al.*, 2015] the Mean Squared Projected Bellman Error (MSPBE):

$$\mathbf{MSPBE}(\theta) = \frac{1}{2} ||\Pi^\pi \mathcal{T}^\pi(\Phi\theta) - \Phi\theta||^2_{D^\pi} \ . \tag{2}$$

By substituting the definition of Π^π into (2), we can write MSPBE as a standard weighted least-squares problem (See [Sutton *et al.*, 2009] for a complete derivation):

$$\mathbf{MSPBE}(\theta) = \frac{1}{2} \|A\theta - b\|^2_{C^{-1}} \tag{3}$$

where A, b and C are defined as follows:

$$A = \mathbb{E}[\phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T] = \Phi^\top D^\pi (I - \gamma P^\pi)\Phi$$
$$b = \mathbb{E}[\phi(s_t) r_t] = \Phi^\top D^\pi r^\pi$$
$$C = \mathbb{E}[\phi(s_t)\phi(s_t)^T] = \Phi^\top D^\pi \Phi$$

where the expectations are taken with respect to the stationary distribution.

Empirical MSPBE: We focus here on the batch setting where we collect a dataset of n transitions $\{(s_t, a_t, r_t, s_{t+1})\}_{t \in [n]}$ generated by the policy π. We replace the quantities A, b and C in (3) by their empirical estimates:

$$\hat{A} = \frac{1}{n} \sum_{t=1}^{n} \hat{A}_t, \quad \hat{b} = \frac{1}{n} \sum_{t=1}^{n} \hat{b}_t, \quad \hat{C} = \frac{1}{n} \sum_{t=1}^{n} \hat{C}_t. \tag{4}$$

where for all $t \in [n]$, for a given transition (s_t, a_t, r_t, s_{t+1})

$$\hat{A}_t \triangleq \phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^\top, \hat{b}_t \triangleq r_t \phi(s_t),$$
$$\hat{C}_t \triangleq \phi(s_t)\phi(s_t)^\top \tag{5}$$

Therefore we consider the empirical MSPBE defined as follows:

$$\textbf{EM-MSPBE}(\theta) = \frac{1}{2} \left\| \hat{A}\theta - \hat{b} \right\|^2_{\hat{C}^{-1}} \tag{6}$$

Finite sum structure: We aim at using stochastic variance-reduction techniques to our problem. These methods are designed for problem with finite sum structure as follows:

$$\min_x f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{7}$$

Unfortunately, even by replacing quantities A, b and C by their finite-sample estimates, the obtained empirical objective in (6) could not be written in such form (7). However, Du et al. [2017] convert the empirical MSPBE minimization in (6) into a convex-concave saddle point problem which presents a finite sum structure. To this end, Du et al. [2017] use the convex-conjugate trick, and the empirical MSPBE minimization is shown to be equivalent to:

$$\min_\theta \max_\omega \left(f(\theta, \omega) = \langle \hat{b} - \hat{A}\theta, \omega \rangle - \frac{1}{2} ||\omega||^2_{\hat{C}} \right) \tag{8}$$

The obtained objective, we denote by $f(\theta, \omega)$, in (8) could be written as $f(\theta, \omega) = \frac{1}{n} \sum_{t=1}^{n} f_t(\theta, \omega)$ where $f_t(\theta, \omega) = \langle \hat{b}_t - \hat{A}_t\theta, \omega \rangle - \frac{1}{2} ||\omega||^2_{\hat{C}_t}$

4 Existing Optimization Algorithms

Before presenting our new methods, we first review briefly existing algorithms that solve the saddle-point problem (8). Let's define the vector $F(\theta, \omega)$ obtained by stacking the primal and negative dual gradients:

$$F(\theta, \omega) \triangleq \begin{pmatrix} \nabla_\theta f(\theta, \omega) \\ -\nabla_\omega f(\theta, \omega) \end{pmatrix} = \begin{pmatrix} -\hat{A}^\top \omega \\ \hat{A}\theta - \hat{b} + \hat{C}\omega \end{pmatrix} \tag{9}$$

We have $F(\theta, \omega) = \frac{1}{n} \sum_{t=1}^{n} F_t(\theta, \omega)$ where $\forall t \in [n]$: $F_t(\theta, \omega) \triangleq \begin{pmatrix} -\hat{A}_t^\top \omega \\ \hat{A}_t\theta - \hat{b}_t + \hat{C}_t\omega \end{pmatrix}$

Gradient temporal difference: GTD2 algorithm [Sutton *et al.*, 2009], when applied to the batch setting, consists of the following update: for a uniformly sampled $t \in [n]$:

$$\begin{pmatrix} \theta \\ \omega \end{pmatrix} \leftarrow \begin{pmatrix} \theta \\ \omega \end{pmatrix} - \begin{pmatrix} \sigma_\theta & 0 \\ 0 & \sigma_\omega \end{pmatrix} F_t(\theta, \omega) \tag{10}$$

where σ_θ and σ_ω are step sizes on θ and ω. GTD2 has a low computation cost per iteration but only a sublinear convergence rate [Touati *et al.*, 2018].

SVRG for policy evaluation: Du et al. [2017] applied SVRG to solve the saddle-point problem (8). The idea is to alternate between full and stochastic gradient updates in two layers of loops. In the outer loop, a snapshot $(\tilde{\theta}, \tilde{\omega})$ of the current variables is saved together with its full gradients vector $F(\tilde{\theta}, \tilde{\omega})$. Between snapshots, the variables are updated with a gradient estimate corrected using the stochastic gradient:

$$v_t \triangleq F_t(\theta, \omega) + F(\tilde{\theta}, \tilde{\omega}) - F_t(\tilde{\theta}, \tilde{\omega}) \tag{11}$$

where $t \in [n]$ is uniformly sampled. Du et al. [2017] showed that the algorithm has a linear convergence rate although the objective (8) is not strongly convex in the primal variable θ. However, the algorithm remains inefficient in term of computations as it requires to compute a full gradient using the entire dataset in the outer loop. In the rest of the paper, "an epoch" means an iteration of the outer loop. In the sequel, we introduce two variants of SVRG for policy evaluation that alleviate the computational bottleneck in SVRG while preserving the linear convergence rate.

5 Batching SVRG and SCSG for Policy Evaluation

Both batching SVRG [Harikandeh *et al.*, 2015] and SCSG [Lei and Jordan, 2017] were initially introduced for convex minimization. Here, we apply them to our convex-concave saddle-point objective function. Algorithm 1 shows the pseudo-code of our methods.

Batching SVRG for policy evaluation estimates full primal and dual gradients in each epoch m using only a subset $\mathcal{B}$ (a mini-batch) of size $|\mathcal{B}| = B_m$ of training examples: $\mu_m = \frac{1}{B_m}\sum_{t\in\mathcal{B}_m} F_t(\tilde{\theta}, \tilde{\omega})$. In each iteration j of the inner loop, it uses $v_{m,j}$ to update θ and ω. $v_{m,j}$ is the usual SVRG update, except that the full gradients is replaced with the mini-batch gradients μ_m. $v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu_m$ where t_j is sampled uniformly in $[n]$.

SCSG for policy evaluation implements the gradient computation on a subset $\mathcal{B}$ of training examples at each epoch. Unlike batching SVRG, the mini-batch size is fixed in advance and not varying. Moreover, instead of being fixed, the number of iteration for the inner loop in SCSG is sampled from a geometrically distributed random variable: $K_m \sim \text{Geom}(\frac{B}{B+1})$ for each epoch m.

6 Convergence Analysis

6.1 Notations and Preliminary

In order to characterize the convergence rates of proposed algorithms, we need to introduce some new notations and state new assumptions.

We denote by $\|A\| \triangleq \sup_{\|x\|=1} \|Ax\|$ the spectral norm of the matrix A and by $\kappa(A) = \|A\|\|A^{-1}\|$ its condition number. If the eigenvalues of a matrix A are real, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote respectively the largest and the smallest eigenvalue.

If we set $\sigma_\omega = \beta\sigma_\theta$ for a positive constant β, it is possible to write the inner loop update (line 11) as an update for the vector $z_{m,j} \triangleq \begin{pmatrix} \theta_{m,j} \\ \frac{1}{\sqrt{\beta}}\omega_{m,j} \end{pmatrix}$ as follows :

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G_m - G_{t_j}) z_{m,0} - g_m \right)$$

where:

$$G_t \triangleq \begin{pmatrix} 0 & -\sqrt{\beta}\hat{A}_t^\top \\ \sqrt{\beta}\hat{A}_t & \beta\hat{C}_t \end{pmatrix}, \quad g_t \triangleq \begin{pmatrix} 0 \\ \sqrt{\beta}\hat{b}_t \end{pmatrix}$$

and their corresponding averages over the mini-batch $\mathcal{B}_m$:

$$G_m \triangleq \frac{1}{|\mathcal{B}_m|}\sum_{t\in\mathcal{B}_m} G_t, \quad g_m \triangleq \frac{1}{|\mathcal{B}_m|}\sum_{t\in\mathcal{B}_m} g_t$$

Let's now define the matrix G (the vector g) as the average of matrices G_t (vectors g_t) over the entire dataset:

$$G \triangleq \begin{pmatrix} 0 & -\sqrt{\beta}\hat{A}^\top \\ \sqrt{\beta}\hat{A} & \beta\hat{C} \end{pmatrix} \quad \text{and} \quad g \triangleq \begin{pmatrix} 0 \\ \sqrt{\beta}\hat{b} \end{pmatrix}$$

To simplify notations, we overload the notation $\lambda_{\min} = \lambda_{\min}(G)$. Another important quantity that characterizes *smoothness* of our problem is L_G^2 defined below as:

$$L_G^2 = \left\| \frac{1}{n}\sum_{t=1}^{n} G_t^\top G_t \right\| \tag{12}$$

The matrix G will play a key role in the convergence analysis of both batching SVRG and SCSG. Du et al. [2017] have already studied the spectral properties of G as it was critical for the convergence of SVRG for policy evaluation. The

Algorithm 1 Batching SVRG and SCSG for policy evaluation. Line 3 and 6 are steps only in batching SVRG. Line 4 and 7 are steps only in SCSG.

Batching SVRG's input: initial point (θ, ω), $\sigma_\theta, \sigma_\omega$, M, K
SCSG's input: initial point (θ, ω), $\sigma_\theta, \sigma_\omega$, M, K and B
Output: (θ, ω)

1: **for** m = 0 to M-1 **do**
2: Set $(\tilde{\theta}, \tilde{\omega})$ and $(\theta_{m,0}, \omega_{m,0})$ to (θ, ω).
3: Batching SVRG's step: choose a mini-batch size B_m. Sample a set $\mathcal{B}$ with B_m elements uniformly from $[n]$
4: SCSG's step: sample a set $\mathcal{B}$ with B elements uniformly from $[n]$
5: Compute $\mu_m = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} F_t(\tilde{\theta}, \tilde{\omega})$
6: Batching SVRG's step: $K_m = K$
7: SCSG's step: $K_m \sim \text{Geom}(\frac{B}{B+1})$
8: **for** j = 0 to $K_m - 1$ **do**
9: Sample t_j uniformly randomly from $[n]$
10: $v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu_m$
11:

$$\begin{pmatrix} \theta_{m,j+1} \\ \omega_{m,j+1} \end{pmatrix} \leftarrow \begin{pmatrix} \theta_{m,j} \\ \omega_{m,j} \end{pmatrix} - \begin{pmatrix} \sigma_\theta & 0 \\ 0 & \sigma_\omega \end{pmatrix} v_{m,j}$$

12: **end for**
13: $(\theta, \omega) = (\theta_{m,K_m}, \omega_{m,K_m})$
14: **end for**
15: **return** (θ, ω)

following lemma, restated from [Du *et al.*, 2017], shows the condition β should satisfy so that G is diagonalizable with all its eigenvalues real and positive.

Assumption 1. *$\hat{A}$ nonsingular and $\hat{C}$ is definite positive. This implies that the saddle-point problem admits a unique solution $(\theta^\star, \omega^\star) = (\hat{A}^{-1}\hat{b}, 0)$ and we define $z^\star \triangleq (\theta^\star, \frac{1}{\sqrt{\beta}}\omega^\star)$.*

Lemma 1. *[Du* et al.*, 2017] Suppose assumption 1 holds and if we choose $\beta = \frac{\sigma_\omega}{\sigma_\theta} = \frac{8\lambda_{\max}(\hat{A}^T\hat{C}^{-1}\hat{A})}{\lambda_{\min}(\hat{C})}$, then the matrix G is diagonalizable with all its eigenvalues real and positive.*

If assumptions of lemma 1 hold, we can write G as $G = Q\Lambda Q^{-1}$ where Λ is a diagonal matrix whose diagonal entries are the eigenvalues of G, and Q consists of it eigenvectors as column. We define the residual vector $\Delta_m = z_m - z^\star$. To study the behaviour of our algorithms, we use the potential function $\left\|Q^{-1}\Delta_m\right\|^2$. As $\|Q\|^2 \left\|Q^{-1}\Delta_m\right\|^2 \geq \|\Delta_m\|^2 \geq \|\theta - \theta^\star\|^2$, the convergence of $\left\|Q^{-1}\Delta_m\right\|^2$ implies the convergence of $\|\theta - \theta^\star\|^2$.

6.2 Convergence of Batching SVRG for Policy Evaluation

In order to study the behavior of batching SVRG for policy evaluation, we defined e_m as the error occurred at epoch m. This error comes from computing the gradients over a mini-batch $\mathcal{B}_m$ instead of the entire dataset.

$$e_m = (Gz_m - g) - (G_m z_m - g_m) \tag{13}$$

The stochastic update of the inner loop could be written as follows:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G - G_{t_j})z_m - g - e_m\right) \tag{14}$$

Theorem 1. *Suppose assumption 1 holds and if we choose $\sigma_\theta = \frac{\lambda_{\min}}{6\kappa(Q)^2 L_G^2}$, $\beta = \frac{8\lambda_{\max}(\hat{A}^T\hat{C}^{-1}\hat{A})}{\lambda_{\min}(\hat{C})}, \sigma_\omega = \beta\sigma_\theta$ and $K = \frac{2}{\sigma_\theta\lambda_{\min}} = \frac{12\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}$, then $\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2$ is upper bounded by:*

$$\underbrace{\frac{2}{3}\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2}_{\text{linear convergence term}} + \underbrace{2\frac{1-\sigma_\theta\lambda_{\min}}{\lambda_{\min}^2}\mathbb{E}\left\|Q^{-1}e_m\right\|^2}_{\text{additional error term}} \tag{15}$$

Full proof of theorem 1 is in [Peng *et al.*, 2019].

Note that if $B_m = n \quad \forall m$, the error is zero and we recover the convergence rate of SVRG in theorem 1. Moreover, we could still maintain the linear convergence rate if the error term vanishes at an appropriate rate. In particular, the corollary below provides a possible batching strategy to control the error term.

Corollary 1. *Suppose that assumptions of theorem 1 hold. If the sample variance of the norms of the vectors F_t is bounded by a constant Ξ^2: $\frac{1}{n-1}\sum_{t=1}^n \|G_t z - g_t\|^2 - \|Gz - g\|^2 \leq \Xi^2 \quad \forall z$ and we set $B_m = \frac{n\Xi^2}{\Xi^2 + n\alpha\rho^m} < n$ for some constants $\alpha > 0$ and $\rho < 2/3$ then $\mathbb{E}\left\|Q^{-1}\Delta_M\right\|^2$ is upper bounded by:*

$$\left(\frac{2}{3}\right)^M \left(\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{3\alpha(1-\sigma_\theta\lambda_{\min})}{\lambda_{\min}^2(1-3\rho/2)}\|Q^{-1}\|^2\right)$$

We conclude that an exponentially-increasing schedule of mini-batch sizes achieves linear convergence rate for batching SVRG. This batching strategy saves many gradient computations in early stages of the algorithm comparing to vanilla SVRG.

6.3 Convergence of SCSG for Policy Evaluation

Before stating the convergence result, we introduce *the complexity measure* defined as follows:

$$\mathcal{H} = \frac{1}{n}\sum_{t=1}^n \|G_t z^\star - g_t\|^2 \tag{16}$$

This quantity is equivalent to the complexity measure that is introduced by [Lei and Jordan, 2017] to analyze SCSG for convex minimization problems, and is defined as $\mathcal{H}(f) = \inf_{x^\star \in \arg\min f(x)} \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*)\|^2$.

Theorem 2. *Suppose assumption 1 holds. Set $\sigma_\theta \leq \min\{\frac{\lambda_{min}}{20\kappa(Q)^2 L_G^2}, \frac{5}{28B\lambda_{max}}\}$, $\sigma_\omega = \beta\sigma_\theta$ and $B \geq \frac{70\kappa(Q)^2 L_G^2}{\lambda_{min}^2}$. The $\mathbb{E}\|\theta_{M-1} - \theta^\star\|^2$ is upper bounded by:*

$$\frac{(1+0.7\sigma_\theta B\lambda_{max})\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\|\Delta_0\|^2 + \frac{60\kappa(Q)^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \tag{17}$$

Full proof of theorem 2 is in [Peng *et al.*, 2019].

Proof sketch: We first take the squared two norms and expectations on SCSG's update, then use the property of geometric random variables. We have:

$$\begin{aligned}
&2\sigma_\theta B\,\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m+1}, Q^{-1}\Delta_{m+1}\right\rangle + \mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \\
&\leq \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta B\,\mathbb{E}\left\langle Q^{-1}e_m, Q^{-1}\Delta_{m+1}\right\rangle \\
&+ 2B\sigma_\theta^2\kappa(Q)^2 L_G^2\left(\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 + \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2\right) \\
&+ 2B\sigma_\theta^2\,\mathbb{E}\left\|\Lambda Q^{-1}\Delta_{m+1}\right\|^2 + 2B\sigma_\theta^2\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{18}$$

Since assumption 1 holds, we can derive two inequalities by applying lemma 1: $\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq \frac{1}{\lambda_{\min}}\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}$ and $\mathbb{E}\left\|\Lambda Q^{-1}\Delta_{m+1}\right\|^2 \leq \lambda_{\max}\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m+1}, Q^{-1}\Delta_{m+1}\right\rangle$ where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of Λ. Moreover, using the technical lemma B.1 in [Lei and Jordan, 2017], we show that $\mathbb{E}\left\|Q^{-1}e_m\right\|^2$ is upper bounded by:

$$\frac{2I(B<n)}{B}\kappa(Q)^2 L_G^2\,\mathbb{E}\,\|Q^{-1}\Delta_m\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H}$$

Plugging theses bounds to (18), we arrive at:

$$\begin{aligned}
&c_1\left(\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 + c_2\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}\right) \\
&\leq \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + c_2\,\mathbb{E}\,\Delta_m^\top Q^{-\top}\Lambda Q^{-1}\Delta_m \\
&+ \frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H}
\end{aligned} \tag{19}$$

where $c_1 = (1 + 0.8\sigma_\theta B\lambda_{\min})$ and $c_2 = 0.7\sigma_\theta B$. Values of c_1 and c_2 come from our settings of σ_θ and B. We prove the theorem after enrolling the above inequality from $m = 0$ to $M - 1$ and doing some algebraic manipulations.

Corollary 2. *Suppose assumption 1 holds. Set* $\sigma_\theta = \min\left\{\frac{\lambda_{min}}{20\kappa(Q)^2 L_G^2}, \frac{5}{28B\lambda_{max}}\right\}$, $\sigma_\omega = \beta\sigma_\theta$ *and* $B = \min\left\{\max\left\{\frac{70\kappa(Q)^2 L_G^2}{\lambda_{min}^2}, \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}\right\}, n\right\}$. *Let* $\epsilon > 0$, *the computational cost in expectations that SCSG required to obtain* $\mathbb{E}\left\|\theta_{M-1} - \theta^\star\right\|^2 \leq \epsilon$ *is:*

$$O\left(\left(\min\left\{\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}, n\right\} + \frac{\kappa(Q)^2 L_G^2}{\lambda_{min}^2}\right) d \times \log\left(\frac{\kappa(Q)^2\mathbb{E}\|\Delta_0\|^2}{\epsilon}\right)\right)$$

Table 1 lists computational costs of our methods and other related methods. GTD2 is the cheapest computationally but it has a sublinear convergence rate. Both SVRG and SCSG achieve linear convergence rates. When the dataset size n is small, SVRG and SCSG have an equivalent computational cost. However, when n is large and the required accuracy ϵ is low, SCSG saves unnecessary computations and is able to achieve the target accuracy with potentially less than a single pass through the dataset.

We compare the computational costs of batching SVRG and SVRG. In epoch m, batching SVRG's computational cost is $O(B_m + K_m)$, where B_m is the batch size and K_m is the number of inner loop iteration. According to theorem 1, K_m is set as $O(\frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2})$ for all m. If we use an exponentially increasing sequence of batch sizes that we considered in corollary 1, B_m is strictly less than n, for all m. Since batching SVRG converges linearly, it takes $O(\ln(1/\epsilon))$ epochs to reach an ϵ-optimal solution. The overall computational cost of batching SVRG is $O\left(\left(n' + \frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}\right) d \times \ln(1/\epsilon)\right)$ as shown in table 1, where $n' < n$, so batching SVRG is computationally more efficient than the vanilla SVRG.

7 Related Works

GTD2 [Sutton *et al.*, 2009] converges to the TD fixed point with linear function approximation. Although derived using MSPBE, GTD2 actually performs stochastic gradient updates on the saddle point formulation of MSPBE [Liu *et al.*, 2015]. Finite sample analysis [Wang *et al.*, 2017; Touati *et al.*, 2018] and accelerations [Liu *et al.*, 2015; Du *et al.*, 2017] on GTD2 becomes possible by using the saddle point formulation of MSPBE. Du et al. [2017] applied SVRG [Johnson and Zhang, 2013] and SAGA [Defazio *et al.*, 2014] to policy evaluation.

Algorithm	Computational Cost
GTD2	$O\left(\frac{\kappa(Q)^2\mathcal{H}d}{\lambda_{\min}^2\epsilon}\right)$
SAGA	$O\left(\left(n + \frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}\right) d \times \ln(1/\epsilon)\right)$
SVRG	$O\left(\left(n + \frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}\right) d \times \ln(1/\epsilon)\right)$
Batching SVRG	$O\left(\left(n' + \frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}\right) d \times \ln(1/\epsilon)\right)$
SCSG	$O\left(\left(\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{\min}^2\epsilon} \wedge n + \frac{\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}\right) d \times \ln(1/\epsilon)\right)$

Table 1: Computational costs of different gradient based policy evaluation algorithms. We report the computational cost of GTD2 from [Touati *et al.*, 2018] and computational costs of SVRG and SAGA from [Du *et al.*, 2017]. We use quantities in our result to represent their computational costs.

Tasks	SVRG	Batching SVRG
Random MDP	100	**71**
Mountain Car	40	**31**
Cart Pole	100	**71**
Acrobot	100	**71**

Table 2: Computational costs of batching SVRG and SVRG. This table shows the number of passes through the dataset for SVRG and batching SVRG when generating results given in Figure 1.

Both methods can be inefficient when the size of the data set is large. We proposed two variants of SVRG with computational costs independent of the size of the data set, while preserving a linear convergence rate.

In control settings, [Papini *et al.*, 2018] and [Xu *et al.*, 2019] adapted SVRG to policy gradient. Their objectives are non-convex while our objective is a convex-concave. Our methods are policy evaluation algorithms, but they can be applied in control settings by combining with LSTDQ [Lagoudakis and Parr, 2003].

Variance reduction techniques have been applied to temporal difference learning [Korda and L.A., 2015; Xu *et al.*, 2020]. They considered an online setting where the value function is updated as data arrives. We consider a batch setting, where we use the policy to collect a batch of data and then learn the value function.

8 Experiments

Our proposed methods, batching SVRG and SCSG[2], are evaluated with LSTD, SVRG, SAGA and GTD2 on 4 tasks: Random MDP [Dann *et al.*, 2014], MountainCar-v0, CartPole-v1 and Acrobot-v1 [Brockman *et al.*, 2016]. Detailed experiment setups are in [Peng *et al.*, 2019].

8.1 Comparing Batching SVRG and SVRG

We show empirically that batching SVRG converges as fast as SVRG while using less amount of data. Figure 1 shows policy evaluation results of SVRG and batching SVRG in different environments, and table 2 shows computational costs of SVRG and batching SVRG. Given same number of epochs, batching SVRG achieves same level of performances with SVRG while taking fewer passes through the dataset. The empirical performance of batching SVRG is expected because our theoretical result suggests that having an approximation error of the full gradient will not affect the overall convergence speed if the error decreases properly.

8.2 Control Performances

We run gradient based methods and LSTD for policy evaluation and apply the learned policy on control tasks. This lets us test the practicality of our methods. We also intend to test our proposed methods' performances in large datasets. In *Mountain Car (large data), Cart Pole (large data)* and *Acrobot (large data)*, the dataset contains 1 million data samples and each method is only allowed to use the dataset once. We run

[2]Code of our experiments can be found at: https://github.com/zilunpeng/svrg_for_policy_evaluation_with_fewer_gradients

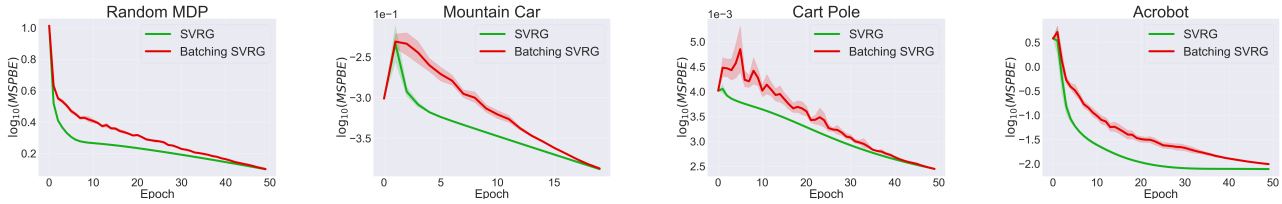

Figure 1: Policy evaluation performances of batching SVRG and SVRG in different environments. Note that batching SVRG and SVRG were run for same number of epochs, and batching SVRG achives same level of performances with SVRG while taking fewer passes through the dataset (shown in table 2).

Task	GTD2	SVRG	SAGA	Batching SVRG	SCSG	LSTD
Mountain Car	348 ± 181	186 ± 149	$\mathbf{166 \pm 125}$	170 ± 135	276 ± 178	292 ± 187
Cart Pole	-163 ± 95	$\mathbf{-280 \pm 88}$	-246 ± 75	$\mathbf{-283 \pm 82}$	-217 ± 100	-183 ± 80
Acrobot	431 ± 128	$\mathbf{96 \pm 6}$	101 ± 6	$\mathbf{97 \pm 6}$	126 ± 78	176 ± 157
Mountain Car (large data)	197 ± 12	200 ± 0	200 ± 0	200 ± 0	199 ± 7	$\mathbf{116 \pm 6}$
Cart Pole (large data)	-155 ± 84	-9 ± 0	-9 ± 0	-297 ± 100	-258 ± 111	$\mathbf{-290 \pm 26}$
Acrobot (large data)	445 ± 96	500 ± 0	500 ± 0	113 ± 26	97 ± 6	$\mathbf{91 \pm 4}$

Table 3: All methods' performances in control. All values in the table are number of steps each learned policy takes to reach the terminal state. Small values mean good performances in control. In Cart Pole setting, large values indicate good performances, so we reported negative values.

all methods with small datasets as well. In *Mountain Car, Cart Pole* and *Acrobot*, the dataset contains 20000 data samples. In all control experiments, we first use a random policy to generate data, then apply policy evaluation algorithms to learn the value function, and finally use the learned policy to control.

Table 3 shows control performances of all methods. We observe that gradient based methods outperform LSTD in experiments when the dataset is small. In large-data experiments, LSTD's performances improve. We run gradient based methods for a single pass through the dataset in large-data experiments. Since SVRG and SAGA need to compute full gradients at the beginning, they cannot solve the control tasks. GTD2, batching SVRG and SCSG do not rely on full gradients, so they make progress instantly. In particular, batching SVRG and SCSG achieve the same level of performance with LSTD in *Cart Pole (large data)* and *Acrobot (large data)*. Our proposed methods and LSTD both use the dataset once and solve the control tasks, while other gradient based methods need more than one pass of the data set. More importantly, unlike LSTD, our methods are first order methods and do not need to invert matrices, which makes our methods practical when both the size of the dataset and the number of state's features are large.

8.3 Policy Evaluation in Large Data Settings

To test our methods' performances on a very large dataset, we generate 10 million data samples from a policy that performs random actions in Random MDP. Figure 2 shows policy evaluation performances of all gradient based methods in Random MDP. We run all methods for a single pass through the dataset. SVRG and SAGA do not make progress because they need to compute full gradients. We observe that batching SVRG and SCSG converge much faster than GTD2, because batching SVRG and SCSG have linear convergence rates, while GTD2 has a sublinear convergence rate. This experiment shows again

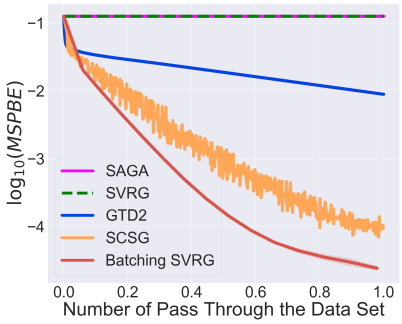

Figure 2: Policy evaluation performances in Random MDP environment within a single pass through the data set.

that our proposed methods have good performances in large data settings.

9 Conclusion

We show that batching SVRG and SCSG converge linearly when solving the saddle-point formulation of MSPBE. This problem is convex-concave and is not strongly convex in the primal variable, so it is very different from the original objective function that batching SVRG and SCSG attempt to solve. Our algorithms are very practical because they require much fewer gradient evaluations than the vanilla SVRG for policy evaluation. There is a lot of room for applying more efficient optimization algorithms to problems in reinforcement learning, in order to obtain better theoretical guarantees and to improve sample and computational efficiency. We think the present work is a valuable contribution in that direction.

References

[Baird, 1995] Leemon Baird. Residual algorithms : Reinforcement learning with function approximation. In *International Conference on Machine Learning*, 1995.

[Balamurugan and Bach, 2016] P. Balamurugan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, 2016.

[Bertsekas, 2011] Dimitri P Bertsekas. Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control*, 2011.

[Bhandari *et al.*, 2018] Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. *Conference on Learning Theory*, 2018.

[Boyan, 2002] Justin Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

[Bradtke and Barto, 1996] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[Dalal *et al.*, 2019] Gal Dalal, Balazs Szorenyi, and Gugan Thoppe. A tale of two-timescale reinforcement learning with the tightest finite-time bound, 2019.

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

[Defazio *et al.*, 2014] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

[Du *et al.*, 2017] Simon S. Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *International Conference on Machine Learning*, 2017.

[Harikandeh *et al.*, 2015] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stop wasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, 2015.

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

[Korda and L.A., 2015] Nathaniel Korda and Prashanth L.A. On td(0) with function approximation: Concentration bounds and a centered variant with exponential convergence. In *International Conference on Machine Learning*, 2015.

[Lagoudakis and Parr, 2003] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, pages 1107–1149, 2003.

[Lei and Jordan, 2017] Lihua Lei and Michael I. Jordan. Less than a single pass: Stochastically controlled stochastic gradient method. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[Liu *et al.*, 2015] Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient td algorithms. In *Conference on Uncertainty in Artificial Intelligence*, 2015.

[Papini *et al.*, 2018] Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. *International Conference on Machine Learning*, 2018.

[Peng *et al.*, 2019] Zilun Peng, Ahmed Touati, Pascal Vincent, and Doina Precup. Svrg for policy evaluation with fewer gradient evaluations. *CoRR*, abs/1906.03704, 2019.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[Sutton *et al.*, 2008] Richard S. Sutton, Cesba Szepesvári, and Hamid Reza Maei. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2008.

[Sutton *et al.*, 2009] Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Cesba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*, pages 993–1000, 2009.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[Touati *et al.*, 2018] Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. Convergent tree backup and retrace with function approximation. In *International Conference on Machine Learning*, pages 4962–4971, 2018.

[Wang *et al.*, 2017] Yue Wang, Wei Chen, Yuting Liu, Zhi-Ming Ma, and Tie-Yan Liu. Finite sample analysis of the gtd policy evaluation algorithms in markov setting. In *Advances in Neural Information Processing Systems*, page 5510–5519, 2017.

[Xu *et al.*, 2019] Pan Xu, Felicia Gao, and Quanquan Gu. An improved convergence analysis of stochastic variance-reduced policy gradient. In *Conference on Uncertainty in Artificial Intelligence*, 2019.

[Xu *et al.*, 2020] Tengyu Xu, Zhe Wang, Yi Zhou, and Yingbin Liang. Reanalysis of variance reduced temporal difference learning. In *International Conference on Learning Representations*, 2020.