Microsoft

A Simple Proximal Stochastic Gradient Method for Nonsmooth Nonconvex Optimization

Zhize Li, **Jian Li**
Tsinghua University

Paper in NIPS18

Problem

We consider the more general nonsmooth nonconvex case:

$$\min_x \Phi(x) := f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x), \tag{1}$$

where each $f_i(x)$ is nonconvex with a Lipschitz continuous gradient ($\exists L$ st. $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$),
while $h(x)$ is nonsmooth (e.g., l_1 regularizer $\|x\|_1$ or indicator function $I_C(x)$ for some constraint set C).

Examples

$$\min_x \Phi(x) := f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x).$$

We list some classical machine learning problems, where $\{x_i, y_i\}_{i=1}^n$ are the training data samples (y_i is the label of data x_i):

Lasso: $f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w) = \frac{1}{n}\sum_i (y_i - w^T x_i)^2$, $h(w) = \|w\|_1$.

l_2-SVM: $f_i(w) = \max\left(0, 1 - y_i(w^T x_i)\right)$, $h(w) = \|w\|_2^2$.

Neural networks:
$f_i(W_k, \ldots, W_1) = \left[\sigma_k\left(W_k \sigma_{k-1}(W_{k-1} \cdots \sigma_1(W_1 x_i) \cdots)\right) - y_i\right]^2$,
$h(w)$ can be a regularization or an indicator function of a constraint set.

Proximal Gradient Descent (ProxGD)

$$\min_{x} \Phi(x) := f(x) + h(x).$$

Replace the GD update as the proximal gradient descent (ProxGD):

$x_t \leftarrow \mathrm{prox}_{\eta h}(x_{t-1} - \eta \nabla f(x_{t-1}))$, for $t \geq 1$,

where $\mathrm{prox}_{\eta h}(x) := \arg\min_{y \in \mathbb{R}^d} \left(h(y) + \frac{1}{2\eta} \|y - x\|^2 \right)$.

- The proximal operator can be viewed as a gradient step on certain "smoothed version" of h (Recall that h may be nonsmooth)
- Prox() is easy to compute for many regularizers

Proximal SGD (ProxSGD)

Recall the problem:

$$\min_x \Phi(x) := f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x).$$

Drawback: GD needs to compute the full gradient in each update step, i.e., $x_t \leftarrow x_{t-1} - \eta\nabla f(x_{t-1}) = x_{t-1} - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x_{t-1})$.

SGD update: randomly choose a subset data samples $\mathcal{I}$, then update $x_t \leftarrow x_{t-1} - \eta\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\nabla f_i(x_{t-1})$.

Note that in the SGD setting, one needs to assume that variance is bounded, i.e., $\mathbb{E}[\|\nabla f_i(x) - \nabla f(x)\|^2] \leq \sigma^2$.

Similarly, ProxSGD update: $\boxed{x_t \leftarrow \mathrm{prox}_{\eta h}\left(x_{t-1} - \eta\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\nabla f_i(x_{t-1})\right).}$

Convergence Criterion

Define the convergence criterion:
$\hat{x}$ is called an ϵ-accurate solution for problem (4) if $\mathbb{E}[\|\mathcal{G}_\eta(\hat{x})\|^2] \leq \epsilon$,
where the gradient mapping $\mathcal{G}_\eta(x) := \frac{1}{\eta}\Big(x - \mathrm{prox}_{\eta h}(x - \eta \nabla f(x))\Big)$.

Recall ProxGD update $x_t \leftarrow \mathrm{prox}_{\eta h}(x_{t-1} - \eta \nabla f(x_{t-1}))$, thus it can be rewritten as $x_t = x_{t-1} - \eta \mathcal{G}_\eta(x_{t-1})$.

Also, $\mathcal{G}_\eta(x) = \nabla \Phi(x) = \nabla f(x)$ if $h(x)$ is a constant function (e.g., 0).
Recall the definition $\mathrm{prox}_{\eta h}(x) := \arg\min_{y \in \mathbb{R}^d} \left(h(y) + \frac{1}{2\eta}\|y - x\|^2\right)$.

Oracle Complexity

To measure the efficiency of a stochastic algorithm for achieving an ϵ-accurate solution, we use the following oracle complexity:

(1) Stochastic first-order oracle (SFO): given a point x, SFO outputs a stochastic gradient $\nabla f_i(x)$ such that $\mathbb{E}_{i\sim[n]}[\nabla f_i(x)] = \nabla f(x)$.

(2) Proximal oracle (PO): given a point x, PO outputs the result of its proximal projection $\mathrm{prox}_{\eta h}(x)$.

For example, consider the ProxGD update:
$x_t \leftarrow \mathrm{prox}_{\eta h}\big(x_{t-1} - \eta \nabla f(x_{t-1})\big) = \mathrm{prox}_{\eta h}\big(x_{t-1} - \eta \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_{t-1})\big)$.
Each update uses n SFO and 1 PO.

Convergence Results of ProxGD and ProxSGD

Theorem ([Ghadimi et al., 2016])

To obtain an ϵ-accurate solution $\hat{x}$ (i.e., $\mathbb{E}[\|\mathcal{G}_\eta(\hat{x})\|^2] \leq \epsilon$), the SFO and PO complexity of ProxGD are $O(\frac{n}{\epsilon})$ and $O(\frac{1}{\epsilon})$ respectively. The SFO and PO complexity of ProxSGD are $O(\frac{b}{\epsilon})$ and $O(\frac{1}{\epsilon})$ respectively, where $b \geq \frac{1}{\epsilon}$ and $\sigma = O(1)$.

Recall that σ comes from the bounded variance assumption, i.e., $\mathbb{E}[\|\nabla f_i(x) - \nabla f(x)\|^2] \leq \sigma^2$.

Original Stochastic Variance Reduced Gradient (SVRG)

Original SVRG by Johnson and Zhang, for convex optimization

To reduce the variance of stochastic gradients

Algorithm 1 Original SVRG

1: $\widetilde{x}^0 = x_0$
2: **for** $s = 1, 2, \ldots$ **do**
3: $\quad x_0^s = \widetilde{x}^{s-1}$
4: $\quad g^s = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\widetilde{x}^{s-1})$
5: $\quad$ **for** $t = 1, 2, \ldots, m$ **do**
6: $\quad\quad$ Randomly pick $i \in [n]$, then compute
$\quad\quad v_{t-1}^s = \nabla f_i(x_{t-1}^s) - \nabla f_i(\widetilde{x}^{s-1}) + g^s$
7: $\quad\quad x_t^s = x_{t-1}^s - \eta v_{t-1}^s$
8: $\quad$ **end for**
9: $\quad$ Option I: $\widetilde{x}^s = x_m^s$
10: $\quad$ Option II: $\widetilde{x}^s = x_{t-1}^s$ for randomly chosen $t \in [m]$
11: **end for**

Unbiased estimation of the gradient, but with much smaller variance

ProxSVRG+ (for nonsmooth nonconvex setting)

$$
\begin{aligned}
&\textbf{for } s = 1, 2, \ldots \textbf{ do}\\
&\quad x_0^s = \widetilde{x}^{s-1}\\
&\quad g^s = \frac{1}{B}\sum_{j \in I_B} \nabla f_j(\widetilde{x}^{s-1})\\
&\quad \textbf{for } t = 1, 2, \ldots, m \textbf{ do}\\
&\qquad v_{t-1}^s = \boxed{\frac{1}{b}\sum_{i \in I_b}} \left(\nabla f_i(x_{t-1}^s) - \nabla f_i(\widetilde{x}^{s-1})\right) + g^s\\
&\qquad x_t^s = \boxed{\mathrm{prox}_{\eta h}}(x_{t-1}^s - \eta v_{t-1}^s)\\
&\quad \textbf{end for}\\
&\quad \widetilde{x}^s = x_m^s
\end{aligned}
$$

Some modification from the ProxSVGG (Reddi et al. NIPS 16)

Our Results

Algorithms	Stochastic first-order orcale (SFO)	Proximal orcale (PO)	Additional condition
ProxGD [Ghadimi et al., 2016] (full gradient)	$O(n/\epsilon)$	$O(1/\epsilon)$	–
ProxSGD [Ghadimi et al., 2016]	$O(b/\epsilon)$	$O(1/\epsilon)$	$\sigma = O(1)$, $b \geq 1/\epsilon$
ProxSVRG/SAGA [Reddi et al., 2016b]	$O(\frac{n}{\epsilon\sqrt{b}} + n)$	$O(\frac{n}{\epsilon b^{3/2}})$	$b \leq n^{2/3}$
SCSG [Lei et al., 2017] (smooth nonconvex, i.e., $h(x) \equiv 0$ in (1))	$O\Big(\frac{b^{1/3}}{\epsilon}(n \wedge \frac{1}{\epsilon})^{2/3}\Big)$	NA	$\sigma = O(1)$
Natasha1.5 [Allen-Zhu, 2017b]	$O(1/\epsilon^{5/3})$ [2]	$O(1/\epsilon^{5/3})$	$\sigma = O(1)$
ProxSVRG+ (this paper)	$O(\frac{n}{\epsilon\sqrt{b}} + \frac{b}{\epsilon})$	$O(1/\epsilon)$	–
	$O\Big((n \wedge \frac{1}{\epsilon})\frac{1}{\epsilon\sqrt{b}} + \frac{b}{\epsilon}\Big)$	$O(1/\epsilon)$	$\sigma = O(1)$

Our Results

Table 2: Some recommended minibatch sizes b

Algorithm	Minibatches	SFO	PO	Addi. cond.	Notes
ProxSVRG+	$b = 1$	$O(n/\epsilon)$	$O(1/\epsilon)$	–	Same as ProxGD
		$O(1/\epsilon^2)$	$O(1/\epsilon)$	$\sigma = O(1)$	Same as ProxSGD
	$b = \frac{1}{\epsilon^{2/3}}$	$O(\frac{n}{\epsilon^{2/3}} + \frac{1}{\epsilon^{5/3}})$	$O(1/\epsilon)$	–	Better than ProxGD, does not need $\sigma = O(1)$
		$O(\frac{1}{\epsilon^{5/3}})$	$O(1/\epsilon)$	$\sigma = O(1)$, $n > 1/\epsilon$	Better than ProxGD and ProxSVRG/SAGA, same as SCSG (in SFO)
	$b = n^{2/3}$	$O(\frac{n^{2/3}}{\epsilon})$	$O(1/\epsilon)$	–	Same as ProxSVRG/SAGA
	$b = n$	$O(n/\epsilon)$	$O(1/\epsilon)$	–	Same as ProxGD

Our Results

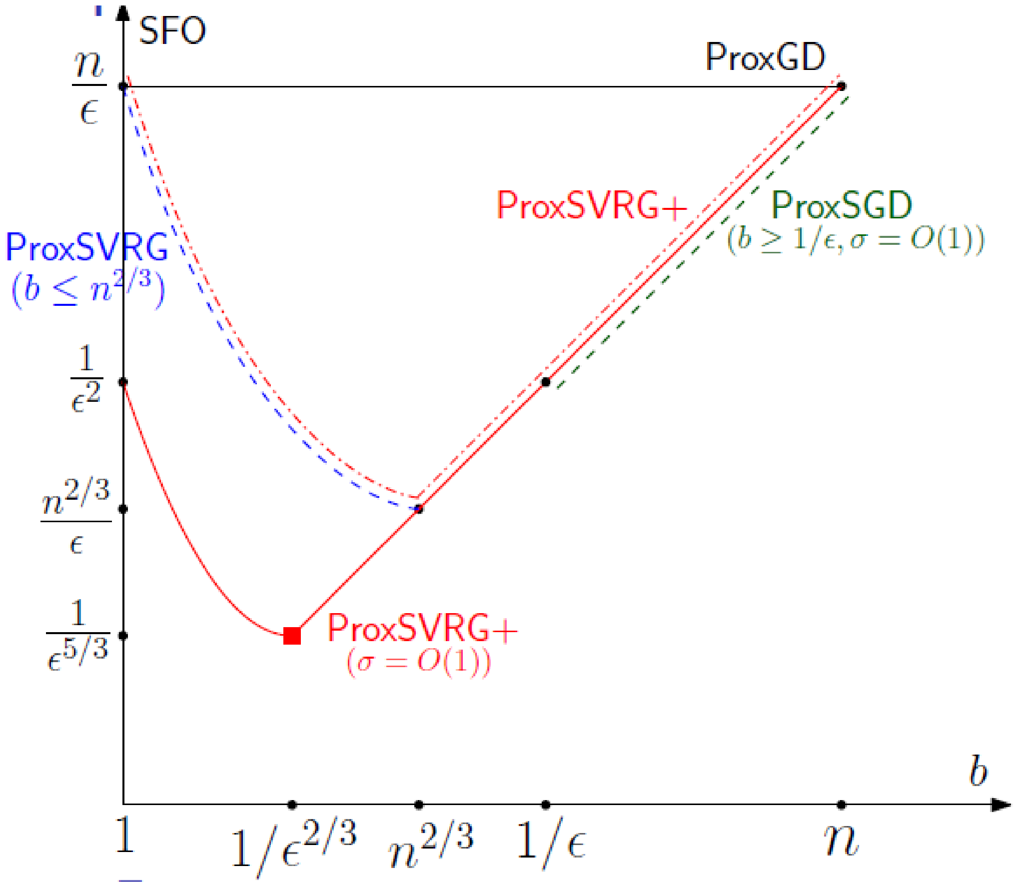

Figure: SFO complexity in terms of b

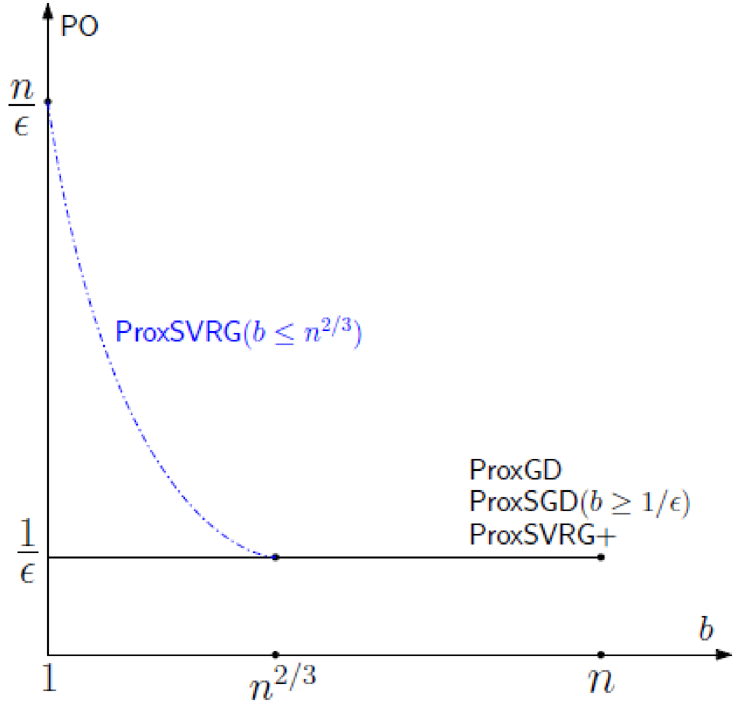

Figure: PO complexity wrt. b

Remarks of Our Results

Our simple ProxSVRG+ algorithm partially answers the open problem proposed by [Reddi et al., 2016] in their ProxSVRG paper:
achieving better performance than ProxGD with constant minibatch size b.

Concretely, ProxSVRG+ is $\sqrt{b}$ times faster than ProxGD when $b \leq n^{2/3}$,
or, ProxSVRG+ is $\sqrt{b\epsilon n}$ times faster than ProxGD when $b \leq 1/\epsilon^{2/3}$.

Remarks of Our Results

Our simple ProxSVRG+ algorithm partially answers the open problem proposed by [Reddi et al., 2016] in their ProxSVRG paper:
achieving better performance than ProxGD with constant minibatch size b.

Concretely, ProxSVRG+ is $\sqrt{b}$ times faster than ProxGD when $b \le n^{2/3}$,
or, ProxSVRG+ is $\sqrt{b\epsilon n}$ times faster than ProxGD when $b \le 1/\epsilon^{2/3}$.

- Our technical contribution mainly lies in our analysis (no time to discuss)
- Our analysis is arguably much simpler than in [Reddi et al. NIPS 16] and [Lei et al. NIPS 17]
- Achieves the best convergence with moderate minibatch size
- In particular, we show the “stochastic controlled” trick is not really necessary in [Lei et al. NIPS 17]

Adapt to Local Convexity

The PL (Polyak-Łojasiewicz) condition [Polyak, 1963]):

$$\exists \mu > 0, \text{ such that } \|\nabla f(x)\|^2 \geq 2\mu\big(f(x) - f(x^*)\big), \ \forall x,$$

For the functions $\Phi(x) = f(x) + h(x)$ satisfying PL condition $(\exists \mu > 0, \ \|\mathcal{G}_\eta(x)\|^2 \geq 2\mu\big(\Phi(x) - \Phi(x^*)\big))$, ProxSVRG+ directly achieves the global linear convergence result $O(\cdot \log \frac{1}{\epsilon})$ instead of the previous $O(\frac{\cdot}{\epsilon})$ for obtaining an ϵ-accurate solution $\hat{x}$ (i.e., $\mathbb{E}[\|\mathcal{G}_\eta(\hat{x})\|^2] \leq \epsilon$).

However, Reddi et al. [2016] used PL-SVRG to restart ProxSVRG $O(\log \frac{1}{\epsilon})$ times for achieving the global linear convergence result.

Experimental results

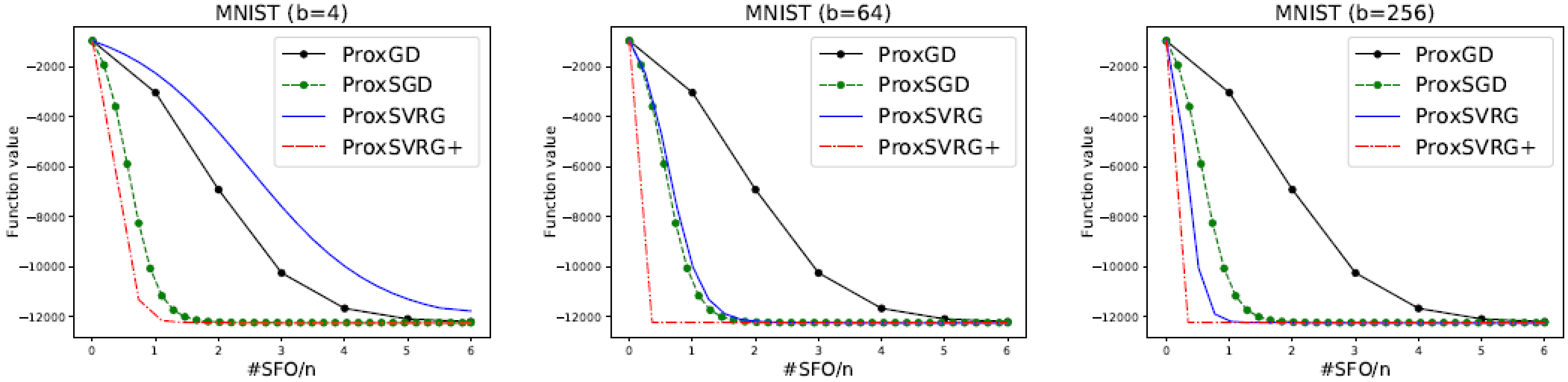

Figure: Comparison among algorithms with different minibatch size b

ProxSVRG+ and ProxSVRG both gets better as b increases.

Experimental Results

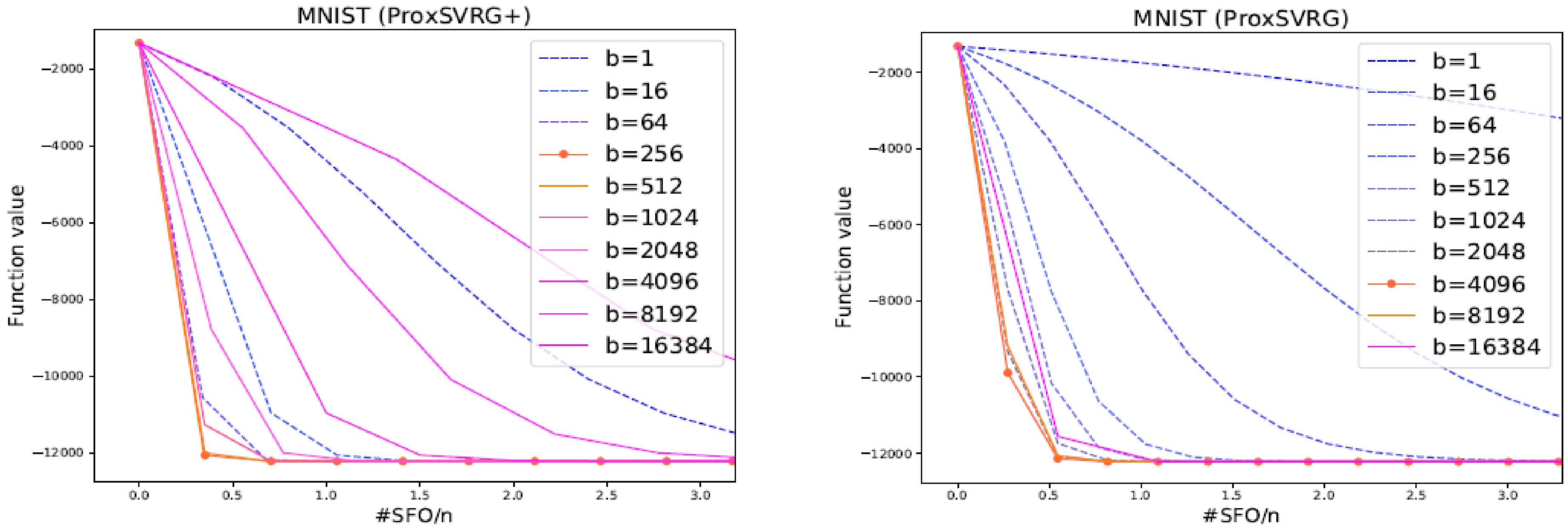

Figure: ProxSVRG+ and ProxSVRG under different minibatch size b

ProxSVRG+ achieves its best performance with smaller b than ProxSVRG.

Concluding Remarks

Finding stationary points -> Finding local min

- Choice: Use Neon2 [Allen-Zhu et al. NIPS18]. Complicated and unnatural.
- Choice: Use perturbation [Jin et al. ICML17]. How to prove the same guarantee.

Very Recently, [Zhou et al. NIPS18] proposed SNVRG.

- Better SFO for smooth case ($n^{1/2}$ instead of $n^{2/3}$)
- Extension to nonsmooth case?

THANKS