Towards Closing the Gap between the Theory and Practice of SVRG

Othmane Sebbouh, Nidham Gazagnadou[a], Samy Jelassi,
Francis Bach, Robert M. Gower

[a]This work was supported by grants from Région Ile-de-France

The Now Famous SVRG Method

- **Finite Sum Minimization problem**

$$x^* = \arg\min_{x \in \mathbb{R}^d} \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \qquad (\mathcal{P})$$

  - f is L-smooth and μ-strongly convex
  - each f_i is $L_{\max}$-smooth

[1] Johnson, Zhang, NIPS, 2013

[2] https://contrib.scikit-learn.org/lightning/

The Now Famous SVRG Method

- **Finite Sum Minimization problem**

$$x^* = \arg\min_{x \in \mathbb{R}^d} \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \qquad (\mathcal{P})$$

  - f is L-smooth and μ-strongly convex
  - each f_i is $L_{\max}$-smooth

- **SVRG**[1] update of the iterates x_s^t

$$\boldsymbol{x_s^{t+1} = x_s^t - \alpha \left(\nabla f_i(x_s^t) - \nabla f_i(w_{s-1}) + \nabla f(w_{s-1}) \right)} \quad ,$$

  where w_{s-1} is a reference point (or snapshot) and i an index randomly sampled in $[n] := \{1, \ldots, n\}$.

[1] Johnson, Zhang, NIPS, 2013

[2] https://contrib.scikit-learn.org/lightning/

The Now Famous SVRG Method

- **Finite Sum Minimization problem**

$$x^* = \arg\min_{x \in \mathbb{R}^d} \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \qquad (\mathcal{P})$$

  - f is L-smooth and μ-strongly convex
  - each f_i is $L_{\max}$-smooth

- **SVRG**[1] update of the iterates x_s^t

$$\boldsymbol{x_s^{t+1} = x_s^t - \alpha \left(\nabla f_i(x_s^t) - \nabla f_i(w_{s-1}) + \nabla f(w_{s-1}) \right)} \quad ,$$

  where w_{s-1} is a reference point (or snapshot) and i an index randomly sampled in $[n] := \{1, \ldots, n\}$.

- **First variance-reduced stochastic method**
  Faster convergence for ERM due to better gradient estimate

[1] Johnson, Zhang, NIPS, 2013
[2] https://contrib.scikit-learn.org/lightning/

The Now Famous SVRG Method

- **Finite Sum Minimization problem**

$$x^* = \underset{x \in \mathbb{R}^d}{\arg\min} \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \qquad (\mathcal{P})$$

  - f is L-smooth and μ-strongly convex
  - each f_i is $L_{\max}$-smooth

- **SVRG**[1] update of the iterates x_s^t

$$\boldsymbol{x_s^{t+1} = x_s^t - \alpha \left(\nabla f_i(x_s^t) - \nabla f_i(w_{s-1}) + \nabla f(w_{s-1}) \right)} \quad ,$$

  where w_{s-1} is a reference point (or snapshot) and i an index randomly sampled in $[n] := \{1, \ldots, n\}$.

- **First variance-reduced stochastic method**
  Faster convergence for ERM due to better gradient estimate
- **Used a lot in practice**
  e.g., adaptations in RL for policy evaluation, implemented in some packages like *lightning*[2]

[1] Johnson, Zhang, NIPS, 2013
[2] https://contrib.scikit-learn.org/lightning/

The Now Famous SVRG Method

- **Finite Sum Minimization problem**

$$x^* = \underset{x \in \mathbb{R}^d}{\arg\min} \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \qquad (\mathcal{P})$$

  - f is L-smooth and μ-strongly convex
  - each f_i is $L_{\max}$-smooth

- **SVRG**[1] update of the iterates x_s^t

$$\boldsymbol{x_s^{t+1} = x_s^t - \alpha \left(\nabla f_i(x_s^t) - \nabla f_i(w_{s-1}) + \nabla f(w_{s-1}) \right)} \quad ,$$

  where w_{s-1} is a reference point (or snapshot) and i an index randomly sampled in $[n] := \{1, \ldots, n\}$.

- **First variance-reduced stochastic method**
  Faster convergence for ERM due to better gradient estimate
- **Used a lot in practice**
  e.g., adaptations in RL for policy evaluation, implemented in some packages like *lightning*[2]

$\rightarrow$ **In practice, implementations differ from the theory**

[1] Johnson, Zhang, NIPS, 2013
[2] https://contrib.scikit-learn.org/lightning/

A Closer Look at SVRG

Algorithm 1 *SVRG* (Johnson, Zhang 2013) & (Bubeck, 2015)

Parameters: $\boldsymbol{m \gtrapprox \frac{L_{\max}}{\mu}}$, ▷ constrained inner loop size

step size α, $p_t := \frac{1}{m}$

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

  $\boldsymbol{x_s^0 = w_{s-1}}$ ▷ resetting the inner iterates to an average

  for $t = 0, 1, \ldots, m-1$ **do**

    Sample i_t uniformly at random in $\{1, \ldots, n\}$

    $g_s^t = \nabla f_{\boldsymbol{i_t}}(x_s^t) - \nabla f_{\boldsymbol{i_t}}(w_{s-1}) + \nabla f(w_{s-1})$ ▷ what about mini-batching?

    $x_s^{t+1} = x_s^t - \alpha g_s^t$

  end for

  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

A Closer Look at SVRG

Algorithm 1 *SVRG* (Johnson, Zhang 2013) & (Bubeck, 2015)

Parameters: $\boldsymbol{m \gtrapprox \frac{L_{\max}}{\mu}}$, ▷ constrained inner loop size
step size α, $p_t := \frac{1}{m}$
Initialization: $w_0 = x_0^m \in \mathbb{R}^d$
for $s = 1, 2, \ldots$ **do**
    $\boldsymbol{x_s^0 = w_{s-1}}$ ▷ resetting the inner iterates to an average
    for $t = 0, 1, \ldots, m-1$ **do**
        Sample i_t uniformly at random in $\{1, \ldots, n\}$
        $g_s^t = \nabla f_{\boldsymbol{i_t}}(x_s^t) - \nabla f_{\boldsymbol{i_t}}(w_{s-1}) + \nabla f(w_{s-1})$ ▷ what about mini-batching?
        $x_s^{t+1} = x_s^t - \alpha g_s^t$
    end for
    $w_s = \sum_{t=0}^{m-1} p_t x_s^t$
end for

Several differences with what is done in practice

- Inner loop size **often** set to $\boldsymbol{m = n}$
- Inner iterates are **continuously updated:** $\boldsymbol{x_s^0 = x_{s-1}^m}$
- **Mini-batching** common practice, yet **not clearly explained by theory**

A Closer Look at SVRG

Algorithm 1 *SVRG* (Johnson, Zhang 2013) & (Bubeck, 2015)

Parameters: $\boldsymbol{m \gtrapprox \frac{L_{\max}}{\mu}}$, ▷ constrained inner loop size
step size α, $p_t := \frac{1}{m}$
Initialization: $w_0 = x_0^m \in \mathbb{R}^d$
for $s = 1, 2, \ldots$ **do**
  $\boldsymbol{x_s^0 = w_{s-1}}$ ▷ resetting the inner iterates to an average
  for $t = 0, 1, \ldots, m-1$ **do**
    Sample i_t uniformly at random in $\{1, \ldots, n\}$
    $g_s^t = \nabla f_{i_t}(x_s^t) - \nabla f_{i_t}(w_{s-1}) + \nabla f(w_{s-1})$ ▷ what about mini-batching?
    $x_s^{t+1} = x_s^t - \alpha g_s^t$
  end for
  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$
end for

Several differences with what is done in practice

- Inner loop size **often** set to $\boldsymbol{m = n}$
- Inner iterates are **continuously updated:** $\boldsymbol{x_s^0 = x_{s-1}^m}$
- **Mini-batching** common practice, yet **not clearly explained by theory**

→ Fill the gap between theory and practice of SVRG

Motivations

Goals

- **"Free"** the inner loop size $\boldsymbol{m}$

Motivations

Goals

- **“Free”** the inner loop size $\boldsymbol{m}$
- **Continuously update** the inner iterates $\boldsymbol{x_s^m}$

Motivations

Goals

- **“Free”** the inner loop size $\boldsymbol{m}$
- **Continuously update** the inner iterates $\boldsymbol{x_s^m}$
- Capture **benefits from mini-batching**

Motivations

Goals

- **"Free"** the inner loop size $\boldsymbol{m}$
- **Continuously update** the inner iterates $\boldsymbol{x_s^m}$
- Capture **benefits from mini-batching**

Contributions

- We designed and analyzed two algorithms closer to practice:
  Free-SVRG and *L-SVRG-D*

Motivations

Goals

- **"Free"** the inner loop size $\boldsymbol{m}$
- **Continuously update** the inner iterates $\boldsymbol{x_s^m}$
- Capture **benefits from mini-batching**

Contributions

- We designed and analyzed two algorithms closer to practice:
  Free-SVRG and *L-SVRG-D*
- Our convergence analysis led to optimal inner loop m^* and mini-batch sizes b^*

Motivations

Goals

- **"Free"** the inner loop size $\boldsymbol{m}$
- **Continuously update** the inner iterates $\boldsymbol{x_s^m}$
- Capture **benefits from mini-batching**

Contributions

- We designed and analyzed two algorithms closer to practice:
  Free-SVRG and *L-SVRG-D*
- Our convergence analysis led to optimal inner loop m^* and mini-batch sizes b^*
- Experiments on real data comparing performance of theoretical settings

Talk Overview

Problem reformulation and preliminary results

Free-SVRG

L-SVRG-D

Numerical experiments

Conclusion

Problem reformulation and preliminary results

Stochastic Reformulation of the ERM

- **ERM reformulation**

$$\text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

$$\iff \text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v_i} f_i(x)\right] = \mathbb{E}_{\mathcal{D}}\left[f_{\boldsymbol{v}}(x)\right]$$

where v is a **sampling vector** s.t. $\mathbb{E}_{v\sim\mathcal{D}}\left[v\right] = \mathbb{1}_n$.

Stochastic Reformulation of the ERM

- **ERM reformulation**

$$\text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$
$$\iff \text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v_i} f_i(x)\right] = \mathbb{E}_{\mathcal{D}}\left[f_{\boldsymbol{v}}(x)\right]$$

where v is a **sampling vector** s.t. $\mathbb{E}_{v\sim\mathcal{D}}[v] = \mathbb{1}_n$.

- General SVRG update of the iterates x_s^t

$$x_s^{t+1} = x_s^t - \alpha\left(\nabla f_{\boldsymbol{v}}(x_s^t) - \nabla f_{\boldsymbol{v}}(w_{s-1}) + \nabla f(w_{s-1})\right) \quad ,$$

Stochastic Reformulation of the ERM

- **ERM reformulation**

$$\text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

$$\iff \text{find } x^* \in \arg\min_{x\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v_i} f_i(x)\right] = \mathbb{E}_{\mathcal{D}}\left[f_{\boldsymbol{v}}(x)\right]$$

where v is a **sampling vector** s.t. $\mathbb{E}_{v\sim\mathcal{D}}\left[v\right] = \mathbb{1}_n$.

- General SVRG update of the iterates x_s^t

$$x_s^{t+1} = x_s^t - \alpha\left(\nabla f_{\boldsymbol{v}}(x_s^t) - \nabla f_{\boldsymbol{v}}(w_{s-1}) + \nabla f(w_{s-1})\right) \quad ,$$

- **Arbitrary sampling** includes all types of sampling *e.g.*, **mini-batching without replacement**. Let $S \subset \{1, \ldots, n\}$ be a random set s.t. $\mathbb{P}\left[S = B\right] = 1/\binom{n}{b}$ for all $B \subset \{1, \ldots, n\}, |B| = b$.

$$\text{Let} \quad v_i = \begin{cases} n/b & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

Then, $f_{\boldsymbol{v}}(x) = \frac{1}{b}\sum_{\boldsymbol{i\in S}} f_{\boldsymbol{i}}(x)$ and $\nabla f_{\boldsymbol{v}}(x) = \frac{1}{b}\sum_{\boldsymbol{i\in S}} \nabla f_i(x)$.

Key constant: Expected Smoothness

Recalling that $\nabla f_v(x) = \frac{1}{n}\sum_{i=1}^n v_i \nabla f_i(x)$

Lemma (Expected Smoothness)
Let $v \sim \mathcal{D}$ be a sampling vector. There exists $\mathcal{L} \geq 0$ such that for all $x \in \mathbb{R}^d$,

$$\mathbb{E}_{v\sim\mathcal{D}}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|_2^2\right] \leq 2\mathcal{L}\left(f(x) - f(x^*)\right) \quad .$$

[3] Gower, Loizou, Qian, Sailanbayev, Shulgin, Richtárik (2019), ICML
[4] **Gazagnadou, Gower, Salmon (2019), ICML**

Key constant: Expected Smoothness

Recalling that $\nabla f_v(x) = \frac{1}{n}\sum_{i=1}^{n} v_i \nabla f_i(x)$

Lemma (Expected Smoothness)
Let $v \sim \mathcal{D}$ be a sampling vector. There exists $\mathcal{L} \geq 0$ such that for all $x \in \mathbb{R}^d$,

$$\mathbb{E}_{v\sim\mathcal{D}}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|_2^2\right] \leq 2\boldsymbol{\mathcal{L}}\left(f(x) - f(x^*)\right) \quad .$$

Example: **mini-batching without replacement**

$$\mathcal{L} = \mathcal{L}(\boldsymbol{b}) = \frac{1}{\boldsymbol{b}}\frac{n-\boldsymbol{b}}{n-1}L_{\max} + \frac{n}{\boldsymbol{b}}\frac{\boldsymbol{b}-1}{n-1}L \quad .$$

In particular, $\mathcal{L}(\mathbf{1}) = L_{\max}$ and $\mathcal{L}(\boldsymbol{n}) = L$.

[3] Gower, Loizou, Qian, Sailanbayev, Shulgin, Richtárik (2019), ICML
[4] **Gazagnadou, Gower, Salmon (2019), ICML**

Key constant: Expected Smoothness

Recalling that $\nabla f_v(x) = \frac{1}{n}\sum_{i=1}^{n} v_i \nabla f_i(x)$

Lemma (Expected Smoothness)
Let $v \sim \mathcal{D}$ be a sampling vector. There exists $\mathcal{L} \geq 0$ such that for all $x \in \mathbb{R}^d$,

$$\mathbb{E}_{v\sim\mathcal{D}}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|_2^2\right] \leq 2\boldsymbol{\mathcal{L}}\left(f(x) - f(x^*)\right) \ .$$

Example: **mini-batching without replacement**

$$\mathcal{L} = \mathcal{L}(\boldsymbol{b}) = \frac{1}{\boldsymbol{b}}\frac{n-\boldsymbol{b}}{n-1}L_{\max} + \frac{n}{\boldsymbol{b}}\frac{\boldsymbol{b}-1}{n-1}L \ .$$

In particular, $\mathcal{L}(\mathbf{1}) = L_{\max}$ and $\mathcal{L}(\boldsymbol{n}) = L$.

$\rightarrow$ **$\boldsymbol{\mathcal{L}}$ embodies the complexity of many stochastic algorithms**[3,4]

[3] Gower, Loizou, Qian, Sailanbayev, Shulgin, Richtárik (2019), ICML
[4] **Gazagnadou, Gower, Salmon (2019), ICML**

Talk Overview

Free-SVRG

Our First Variant: Free-SVRG

Algorithm 2 *Free-SVRG* (or 1-SVRG in Raj and Stich, 2018)

Parameters: $\boldsymbol{m}$, ▷ inner loop length freely chosen by the user

step size α, weights $p_t := (1-\alpha\mu)^{m-1-t} \Big/ \sum_{i=0}^{m-1} (1-\alpha\mu)^{m-1-i}$

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

  $\boldsymbol{x_s^0 = x_{s-1}^m}$ ▷ continuous update of the iterates

  for $t = 0, 1, \ldots, m-1$ **do**

    Sample $v_t \sim \mathcal{D}$

    $g_s^t = \nabla f_{v_t}(x_s^t) - \nabla f_{v_t}(w_{s-1}) + \nabla f(w_{s-1})$

    $x_s^{t+1} = x_s^t - \alpha g_s^t$

  end for

  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

Our First Variant: Free-SVRG

Algorithm 2 *Free-SVRG* (or 1-SVRG in Raj and Stich, 2018)

Parameters: $\boldsymbol{m}$, ▷ inner loop length freely chosen by the user

step size α, weights $p_t := (1-\alpha\mu)^{m-1-t} \Big/ \sum_{i=0}^{m-1} (1-\alpha\mu)^{m-1-i}$

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

  $\boldsymbol{x_s^0 = x_{s-1}^m}$ ▷ continuous update of the iterates

  for $t = 0, 1, \ldots, m-1$ **do**

    Sample $v_t \sim \mathcal{D}$

    $g_s^t = \nabla f_{v_t}(x_s^t) - \nabla f_{v_t}(w_{s-1}) + \nabla f(w_{s-1})$

    $x_s^{t+1} = x_s^t - \alpha g_s^t$

  end for

  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

Solves several issues with SVRG

- **Free choice of** $\boldsymbol{m}$ the inner loop size
- Inner iterates $\boldsymbol{x_s^t}$ **continuously updated** (no resetting)
- Analysis **capturing** independently **influence of** $\boldsymbol{m}$ **and** $\boldsymbol{b}$

Convergence Theorem

Theorem (Convergence of Free-SVRG)
Consider the setting of and the following Lyapunov function

$$\phi_s \ := \ \|x_s^m - x^*\|_2^2 + 8\alpha^2 \mathcal{L} S_m (f(w_s) - f(x^*)),$$

where $S_m = \sum_{i=0}^{m-1} (1 - \alpha\mu)^{m-1-i}$.
If $\alpha \leq \frac{1}{6\mathcal{L}}$, *then the iterates of Free-SVRG converge with*

$$\mathbb{E}[\phi_s] \ \leq \ \beta^s \phi_0 \ ,$$

where $\beta := \max\left\{(1 - \alpha\mu)^m, \frac{1}{2}\right\}$.

Optimal Inner Loop Size

- **Total complexity for mini-batching**
  For **mini-batching without replacement**, the **total complexity** of getting an $\epsilon > 0$ approximate solution s.t. $\mathbb{E}\left[\|x_s^m - x^*\|_2^2\right] \leq \epsilon\,\phi_0$ is

$$C_m(\boldsymbol{b}) \;:=\; 2\left(\frac{n}{m} + 2\boldsymbol{b}\right)\max\left\{\frac{3}{\boldsymbol{b}}\frac{n-\boldsymbol{b}}{n-1}\frac{L_{\max}}{\mu} + \frac{3n}{\boldsymbol{b}}\frac{\boldsymbol{b}-1}{n-1}\frac{L}{\mu}, m\right\}\log\left(\frac{1}{\epsilon}\right)$$

Optimal Inner Loop Size

- **Total complexity for mini-batching**
  For **mini-batching without replacement**, the **total complexity** of getting an $\epsilon > 0$ approximate solution s.t. $\mathbb{E}\left[\|x_s^m - x^*\|_2^2\right] \leq \epsilon\, \phi_0$ is

$$C_m(\boldsymbol{b}) \; := \; 2\left(\frac{n}{m} + 2\boldsymbol{b}\right) \max\left\{\frac{3}{\boldsymbol{b}}\frac{n-\boldsymbol{b}}{n-1}\frac{L_{\max}}{\mu} + \frac{3n}{\boldsymbol{b}}\frac{\boldsymbol{b}-1}{n-1}\frac{L}{\mu}, m\right\} \log\left(\frac{1}{\epsilon}\right)$$

- **Optimal inner loop size**
  Let us minimize the total complexity w.r.t. $\boldsymbol{m}$.
  If $m \in \left[\min\left(n, \frac{L_{\max}}{\mu}\right), \max\left(n, \frac{L_{\max}}{\mu}\right)\right]$, then

$$C_m(1) = O\left(\left(\boldsymbol{n} + \frac{\boldsymbol{L_{\max}}}{\boldsymbol{\mu}}\right)\log\left(\frac{1}{\epsilon}\right)\right)$$

Optimal Inner Loop Size

- **Total complexity for mini-batching**
  For **mini-batching without replacement**, the **total complexity** of getting an $\epsilon > 0$ approximate solution s.t. $\mathbb{E}\left[\|x_s^m - x^*\|_2^2\right] \leq \epsilon\,\phi_0$ is

$$C_m(\boldsymbol{b}) \;:=\; 2\left(\frac{n}{m} + 2\boldsymbol{b}\right)\max\left\{\frac{3}{\boldsymbol{b}}\frac{n-\boldsymbol{b}}{n-1}\frac{L_{\max}}{\mu} + \frac{3n}{\boldsymbol{b}}\frac{\boldsymbol{b}-1}{n-1}\frac{L}{\mu}, m\right\}\log\left(\frac{1}{\epsilon}\right)$$

- **Optimal inner loop size**
  Let us minimize the total complexity w.r.t. $\boldsymbol{m}$.
  If $m \in \left[\min\left(n, \frac{L_{\max}}{\mu}\right), \max\left(n, \frac{L_{\max}}{\mu}\right)\right]$, then

$$C_m(1) = O\left(\left(\boldsymbol{n} + \frac{\boldsymbol{L_{\max}}}{\boldsymbol{\mu}}\right)\log\left(\frac{1}{\epsilon}\right)\right)$$

$\rightarrow$ Includes the practical choice $\boldsymbol{m} = \boldsymbol{n}$

Optimal Mini-Batch Size

- For any fixed inner loop size m
  - the **total complexity** is a **convex function** of b
  - the **step size** is an **increasing function** of b

Optimal Mini-Batch Size

- For any fixed inner loop size m
  - the **total complexity** is a **convex function** of b
  - the **step size** is an **increasing function** of b

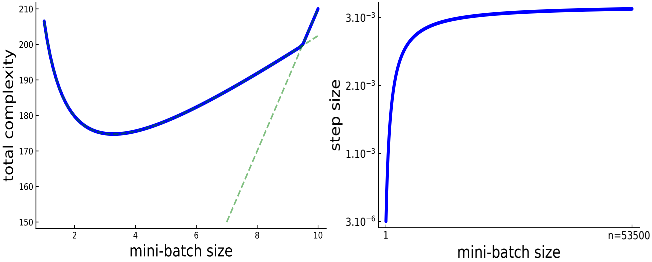

Figure 1: Total complexity (left) and step size (right) as b increases.

Optimal Mini-Batch Size

- For any fixed inner loop size m
  - the **total complexity** is a **convex function** of b
  - the **step size** is an **increasing function** of b

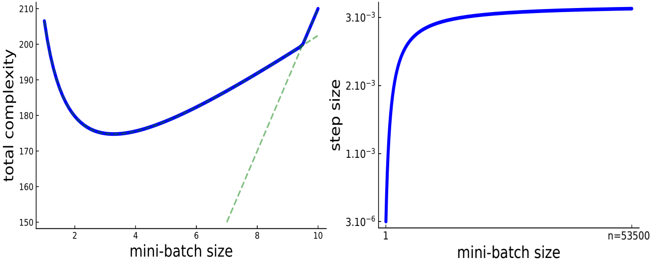

Figure 1: Total complexity (left) and step size (right) as b increases.

- **Optimal mini-batch size** for the usual choice $\boldsymbol{m = n}$

$$b^* = \begin{cases} 1 & \text{if } n \geq \frac{3L_{\max}}{\mu} \\ \left\lfloor \min(\tilde{b}, \hat{b}) \right\rfloor & \text{if } \frac{3L}{\mu} < n < \frac{3L_{\max}}{\mu} \\ \left\lfloor \hat{b} \right\rfloor & \text{otherwise, if } n \leq \frac{3L}{\mu} \end{cases}$$

where

$$\hat{b} := \sqrt{\frac{n}{2} \frac{L_{\max} - L}{nL - L_{\max}}}$$

$$\tilde{b} := \frac{3n(L_{\max} - L)}{n(n-1)\mu - 3(nL - L_{\max})}$$

An Issue with Free-SVRG

Algorithm 2 *Free-SVRG*

Parameters: m, step size α,

weights $p_t := (1-\alpha\boldsymbol{\mu})^{m-1-t} \Big/ \sum_{i=0}^{m-1}(1-\alpha\boldsymbol{\mu})^{m-1-i}$ ▷ need to compute μ first

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

  $x_s^0 = x_{s-1}^m$

  for $t = 0, 1, \ldots, m-1$ **do**

    Sample $v_t \sim \mathcal{D}$

    $g_s^t = \nabla f_{v_t}(x_s^t) - \nabla f_{v_t}(w_{s-1}) + \nabla f(w_{s-1})$

    $x_s^{t+1} = x_s^t - \alpha g_s^t$

  end for

  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

An Issue with Free-SVRG

Algorithm 2 *Free-SVRG*

Parameters: m, step size α,

weights $p_t := (1-\alpha\boldsymbol{\mu})^{m-1-t} \Big/ \sum_{i=0}^{m-1} (1-\alpha\boldsymbol{\mu})^{m-1-i}$ $\triangleright$ need to compute μ first

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

$\quad x_s^0 = x_{s-1}^m$

$\quad$ **for** $t = 0, 1, \ldots, m-1$ **do**

$\quad\quad$ Sample $v_t \sim \mathcal{D}$

$\quad\quad g_s^t = \nabla f_{v_t}(x_s^t) - \nabla f_{v_t}(w_{s-1}) + \nabla f(w_{s-1})$

$\quad\quad x_s^{t+1} = x_s^t - \alpha g_s^t$

$\quad$ **end for**

$\quad w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

Only issue

- *Free-SVRG* **requires the strong convexity** μ often had to estimate

An Issue with Free-SVRG

Algorithm 2 *Free-SVRG*

Parameters: m, step size α,

weights $p_t := (1-\alpha\boldsymbol{\mu})^{m-1-t} \Big/ \sum_{i=0}^{m-1} (1-\alpha\boldsymbol{\mu})^{m-1-i}$ ▷ need to compute μ first

Initialization: $w_0 = x_0^m \in \mathbb{R}^d$

for $s = 1, 2, \ldots$ **do**

  $x_s^0 = x_{s-1}^m$

  for $t = 0, 1, \ldots, m-1$ **do**

    Sample $v_t \sim \mathcal{D}$

    $g_s^t = \nabla f_{v_t}(x_s^t) - \nabla f_{v_t}(w_{s-1}) + \nabla f(w_{s-1})$

    $x_s^{t+1} = x_s^t - \alpha g_s^t$

  end for

  $w_s = \sum_{t=0}^{m-1} p_t x_s^t$

end for

Only issue

- *Free-SVRG* **requires the strong convexity** μ often had to estimate

$\rightarrow$ Other variant: L-SVRG-D

Talk Overview

L-SVRG-D

Our Second Variant: Loopless-SVRG-Decrease

Algorithm 3 *L-SVRG-D*

Parameters: step size α, $p \in (0, 1]$

Initialization: $w^0 = x^0 \in \mathbb{R}^d,\ \alpha_0 = \alpha$

for $k = 0, 1, 2, \ldots$ **do**

Sample $v_k \sim \mathcal{D}$

$g^k = \nabla f_{v_k}(x^k) - \nabla f_{v_k}(w^k) + \nabla f(w^k)$

$x^{k+1} = x^k - \alpha_k g^k$

$$(w^{k+1}, \alpha_{k+1}) = \begin{cases} (x^k, \alpha) & \text{with prob. } p \\ (w^k, \boldsymbol{\sqrt{1-p}}\, \alpha_k) & \text{with prob. } 1 - p \end{cases}$$

end for

Our Second Variant: Loopless-SVRG-Decrease

Algorithm 3 *L-SVRG-D*

Parameters: step size α, $p \in (0,1]$

Initialization: $w^0 = x^0 \in \mathbb{R}^d,\ \alpha_0 = \alpha$

for $k = 0, 1, 2, \ldots$ **do**

Sample $v_k \sim \mathcal{D}$

$g^k = \nabla f_{v_k}(x^k) - \nabla f_{v_k}(w^k) + \nabla f(w^k)$

$x^{k+1} = x^k - \alpha_k g^k$

$$(w^{k+1}, \alpha_{k+1}) = \begin{cases} (x^k, \alpha) & \text{with prob. } p \\ (w^k, \boldsymbol{\sqrt{1-p}}\, \alpha_k) & \text{with prob. } 1-p \end{cases}$$

end for

Benefits

- **Bigger step size** for the first iterations of the loop, when the **variance is low**
- **Smaller step size** for the last iterations of the loop, when the **variance is high**

Our Second Variant: Loopless-SVRG-Decrease

Algorithm 3 *L-SVRG-D*

Parameters: step size α, $p \in (0,1]$

Initialization: $w^0 = x^0 \in \mathbb{R}^d,\ \alpha_0 = \alpha$

for $k = 0, 1, 2, \ldots$ **do**

Sample $v_k \sim \mathcal{D}$

$g^k = \nabla f_{v_k}(x^k) - \nabla f_{v_k}(w^k) + \nabla f(w^k)$

$x^{k+1} = x^k - \alpha_k g^k$

$$(w^{k+1}, \alpha_{k+1}) = \begin{cases} (x^k, \alpha) & \text{with prob. } p \\ (w^k, \sqrt{\mathbf{1-p}}\, \alpha_k) & \text{with prob. } 1-p \end{cases}$$

end for

Benefits

- **Bigger step size** for the first iterations of the loop, when the **variance is low**
- **Smaller step size** for the last iterations of the loop, when the **variance is high**

$\rightarrow$ **Same total complexity and optimal parameter settings as Free-SVRG** (up to constants)

Talk Overview

Problem reformulation and preliminary results

Free-SVRG

L-SVRG-D

Numerical experiments

Conclusion

Numerical experiments

Performance of Theoretical Settings of SVRG Variants

- Data: from LIBSVM and UCI repositories
- Problems: ridge regression and regularized logistic regression, $\lambda \in \{10^{-3}, 10^{-1}\}$

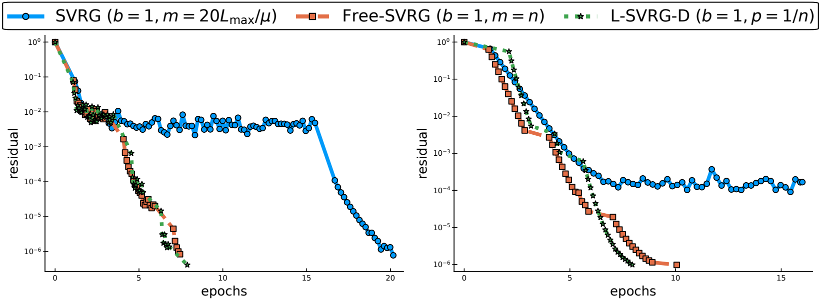

Figure 2: Theoretical settings for SVRG, *Free*-SVRG and *L-SVRG-D*.
Left: l_2-regularized logistic regression on *ijcnn1*.
Right: l_2-regularized ridge regression on *YearPredictionMSD*.

Optimality of our Mini-Batch Size

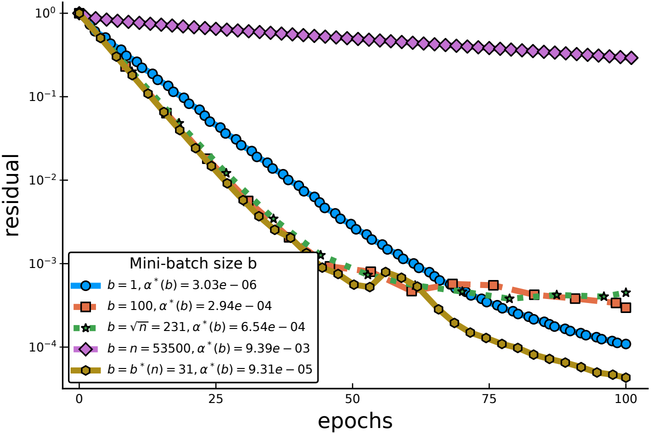

Figure 3: Different mini-batch sizes for *Free-SVRG* for a l_2-regularized ridge regression problem on the *slice* data set.

Optimality of our Inner Loop Size

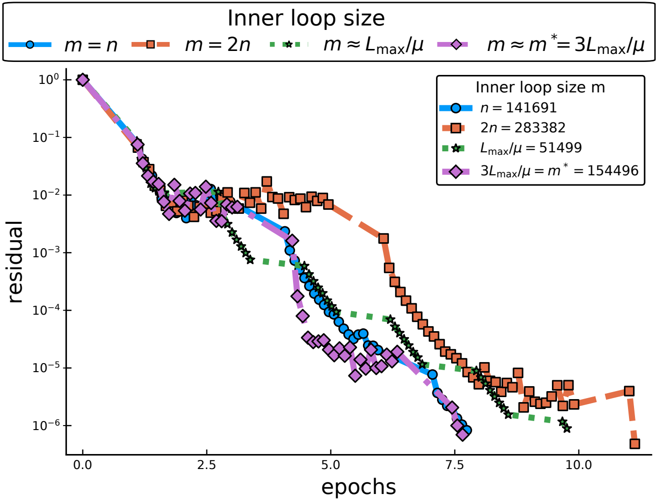

Figure 4: Different inner loop sizes for *Free-SVRG* for a l_2-regularized logistic regression problem on the *ijcnn1* data set.

Talk Overview

Conclusion

Summary of Our Contributions

- Two variants of SVRG designed to get closer to practical implementations: *Free-SVRG* & *L-SVRG-D*

[5] LIBSVM and UCI repositories

Summary of Our Contributions

- Two variants of SVRG designed to get closer to practical implementations: *Free-SVRG* & *L-SVRG-D*
- Optimal parameters: inner loop size m^* and mini-batch size b^*

[5] LIBSVM and UCI repositories

Summary of Our Contributions

- Two variants of SVRG designed to get closer to practical implementations: *Free-SVRG* & *L-SVRG-D*
- Optimal parameters: inner loop size m^* and mini-batch size b^*
- Convincing numerics verifying the performance of our theoretical settings on real data sets[5]

Julia code available at
https://github.com/gowerrobert/StochOpt.jl/

[5] LIBSVM and UCI repositories

Summary of Our Contributions

- Two variants of SVRG designed to get closer to practical implementations: *Free-SVRG & L-SVRG-D*
- Optimal parameters: inner loop size m^* and mini-batch size b^*
- Convincing numerics verifying the performance of our theoretical settings on real data sets[5]

Julia code available at
`https://github.com/gowerrobert/StochOpt.jl/`

More details in
Sebbouh, Gazagnadou, Jelassi, Bach, Gower (2019), NeurIPS
"Towards closing the gap between the theory and practice of SVRG"

[5] LIBSVM and UCI repositories

References

- Chang and Lin (2011), ACM TIST
  "LIBSVM : A library for support vector machines"
- Gazagnadou, Gower and Salmon (2019), ICML
  "Optimal mini-batch and step sizes for SAGA"
- Gower, Loizou, Qian, Sailanbayev, Shulgin, Richtárik (2019), ICML
  "SGD: General Analysis and Improved Rates"
- Gower, Richtárik and Bach (2018), preprint online
  "Stochastic Quasi-Gradient Methods: Variance Reduction via Jacobian Sketching"
- Hofmann, Lucchi, Lacoste-Julien, McWilliams (2015), NIPS
  "Variance Reduced Stochastic Gradient Descent with Neighbors"
- Johnson and Zhang (2013), NIPS
  "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction"
- Raj and Stich (2018), preprint online
  "k-SVRG: Variance Reduction for Large Scale Optimization"
- Sebbouh, Gazagnadou, Jelassi, Bach, Gower (2019), NeurIPS
  "Towards closing the gap between the theory and practice of SVRG"

Thank You!

Questions?