LARGE-SCALE CONVEX OPTIMIZATION: PARALLELIZATION AND VARIANCE REDUCTION

Ph.D. Thesis Defense

June 10th, 2024

Cheik Traoré[1]

Supervisor: **Silvia Villa**[1]

Jury: **Claudio Estatico**[1], **François Glineur**[2] and **Juan Peypouquet**[3]

TRAINING DATA-DRIVEN EXPERTS IN OPTIMIZATION MSCA-ITN 2019

[1]University of Genoa, [2]UCLouvain, [3]University of Groningen

Context

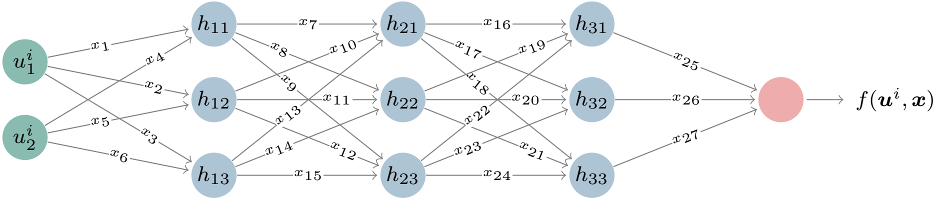
Training dataset: $\left(\boldsymbol{u}^i, y^i\right)_{i \in \{1,2,\cdots,n\}}$.

$$\underset{\boldsymbol{x} \in \mathbb{R}^m}{\text{minimize}} \ \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{u}^i; \boldsymbol{x}), y_i) + \lambda R(\boldsymbol{x}).$$

Context

Training dataset: $(\boldsymbol{u}^i, y^i)_{i \in \{1,2,\cdots,n\}}$.

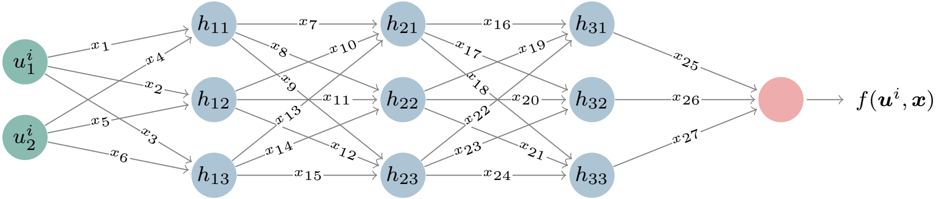
$$\underset{\boldsymbol{x} \in \mathbb{R}^m}{\text{minimize}} \; \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{u}^i; \boldsymbol{x}), y_i) + \lambda R(\boldsymbol{x}).$$



Context

Training dataset: $(\boldsymbol{u}^i, y^i)_{i \in \{1,2,\cdots,n\}}$.

$$\underset{\boldsymbol{x} \in \mathbb{R}^m}{\text{minimize}} \; \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{u}^i; \boldsymbol{x}), y_i) + \lambda R(\boldsymbol{x}).$$

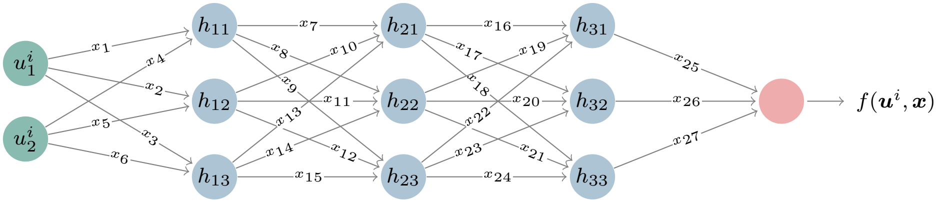


n and m can be very big: several BILLIONS!!!

Context

Training dataset: $\left(\boldsymbol{u}^i, y^i\right)_{i\in\{1,2,\cdots,n\}}$.

$$\underset{\boldsymbol{x}\in\mathbb{R}^m}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n}\ell(f(\boldsymbol{u}^i;\boldsymbol{x}),y_i)+\lambda R(\boldsymbol{x}).$$



n and m can be very big: several BILLIONS!!!

ChatGP4: **estimated** 1.76 trillion parameters *(Georges Hotz)*.

Outline

General introduction

Asynchronous Forward-Backward

Variance reduction techniques for SPPA

Conclusion

Optimization for data science

Goal:

$$\underset{\boldsymbol{x} \in \mathbf{H}}{\text{minimize}}\; F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

where **H** is a separable real Hilbert space and $F, f, g \colon \mathbf{H} \to \mathbb{R}$.

Optimization for data science

Goal:

$$\underset{\boldsymbol{x}\in\mathbf{H}}{\text{minimize}}\; F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

where **H** is a separable real Hilbert space and $F, f, g\colon \mathbf{H} \to \mathbb{R}$.

Example:

$$\underset{\boldsymbol{x}\in\mathbb{R}^m}{\text{minimize}}\; F(\boldsymbol{x}) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})}_{f(\boldsymbol{x})} + \underbrace{\lambda R(\boldsymbol{x})}_{g(\boldsymbol{x})},$$

Most popular method

Gradient descent (GD):

$$\boldsymbol{x}^0 \in \mathbf{H}$$

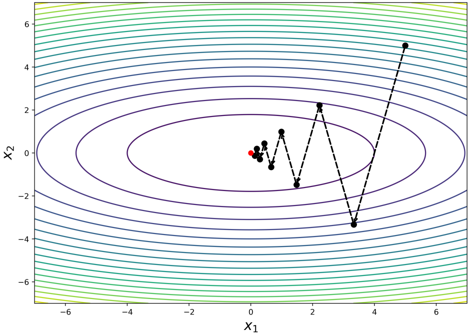
$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla F(\boldsymbol{x}^k).$$

Most popular method

Gradient descent (GD):

$$\boldsymbol{x}^0 \in \mathbf{H}$$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla F(\boldsymbol{x}^k).$$



GD rates

F is $\mu-$strongly convex for $\mu \geq 0$

GD rates

F is $\mu-$strongly convex for $\mu \geq 0$:

$$(\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{H}) \ \ (t \in [0,1]) \quad F(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tF(\boldsymbol{x}) + (1-t)F(\boldsymbol{y}) - \frac{\mu}{2} t(1-t)\|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

GD rates

F is $\mu-$strongly convex for $\mu \geq 0$:

$$(\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{H}) \ \ (t \in [0,1]) \quad F(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tF(\boldsymbol{x}) + (1-t)F(\boldsymbol{y}) - \frac{\mu}{2} t(1-t)\|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

F is $L-$smooth for $L \geq 0$:

$$(\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{H}) \quad \|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

GD rates

Let $\gamma_k = \frac{1}{L}$.

GD rates

Let $\gamma_k = \frac{1}{L}$.

- If $\mu = 0$,

$$F(\boldsymbol{x}_k) - F_* \leq O(1/k) \quad \text{(sublinear rate)}.$$

GD rates

Let $\gamma_k = \frac{1}{L}$.

- If $\mu = 0$,

$$F(\boldsymbol{x}_k) - F_* \leq O(1/k) \quad \text{(sublinear rate)}.$$

- If $\mu > 0$,

$$F(\boldsymbol{x}_k) - F_* \leq O(\varepsilon^k) \quad \text{(linear rate)},$$

  with $\epsilon < 1$.

Sum of functions problem

First case: $g \equiv 0$ and $f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$.

Sum of functions problem

First case: $g \equiv 0$ and $f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$.

The problem is now:

$$\underset{\boldsymbol{x}\in\mathbf{H}}{\text{minimize}}\ F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}).$$

Stochastic gradient descent (SGD)

SGD update is:

Select uniformly at random $i_k \in [n] := \{1, 2, \ldots, n\}$ and do

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}^k).$$

Non-smooth problem

Second case: g is not differentiable.

Non-smooth problem

Second case: g is not differentiable.

The problem is:

$$\operatorname*{minimize}_{\boldsymbol{x} \in \mathbf{H}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

with f smooth.

Non-smooth problem

Second case: g is not differentiable.

The problem is:

$$\underset{\boldsymbol{x} \in \mathbf{H}}{\text{minimize}}\; F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}),$$

with f smooth.

So F is non-smooth!!

Subgradient descent

Update:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \partial F(\boldsymbol{x}^k).$$

Subgradient descent

Update:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \partial F(\boldsymbol{x}^k).$$

Rates with $\gamma_k \to 0$ and bounded subgradient:

- Convex case: $F(\bar{\boldsymbol{x}}^k) - F_* \leq O\left(\frac{1}{\sqrt{k}}\right)$.
- Strongly convex case: $F(\bar{\boldsymbol{x}}^k) - F_* \leq O\left(1/k\right)$.

$\bar{\boldsymbol{x}}^k = \sum_{t=i}^{k} \mathsf{p}_t \boldsymbol{x}^t$ with $\sum_{t=i}^{k} \mathsf{p}_t = 1$ (ergodic rates).

Proximal Point Algorithm (PPA)

Update:

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \text{prox}_{\gamma_k F}(\boldsymbol{x}^k) = \text{prox}_{\gamma_k (f+g)}(\boldsymbol{x}^k) \\ &= \underset{\boldsymbol{x}}{\text{argmin}}\, F(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2 \\ &= \underset{\boldsymbol{x}}{\text{argmin}}\, f(\boldsymbol{x}) + g(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2. \end{aligned}$$

Proximal Point Algorithm (PPA)

Update:

$$
\begin{aligned}
\boldsymbol{x}^{k+1} &= \mathrm{prox}_{\gamma_k F}(\boldsymbol{x}^k) = \mathrm{prox}_{\gamma_k (f+g)}(\boldsymbol{x}^k) \\
&= \operatorname*{argmin}_{\boldsymbol{x}} F(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2 \\
&= \operatorname*{argmin}_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2.
\end{aligned}
$$

Rates with $\gamma_k = \gamma \in \mathbb{R}_+$:

- Convex case: $F(\boldsymbol{x}_k) - F_* \leq O(1/k)$.
- Strongly convex case: $F(\boldsymbol{x}_k) - F_* \leq O(\varepsilon^k)$, with $\epsilon < 1$.

Forward-Backward

Proximal gradient (Forward-Backward) update:

$$\boldsymbol{x}^{k+1} = \mathsf{prox}_{\gamma_k g}\left(\boldsymbol{x}^k - \gamma_k \nabla f(\boldsymbol{x}^k)\right).$$

Forward-Backward

Proximal gradient (Forward-Backward) update:

$$\boldsymbol{x}^{k+1} = \mathrm{prox}_{\gamma_k g}\left(\boldsymbol{x}^k - \gamma_k \nabla f(\boldsymbol{x}^k)\right).$$

Rates with $\gamma_k = \frac{1}{L}$:

- Convex case: $F(\boldsymbol{x}_k) - F_* \leq O(1/k)$.
- Strongly convex case: $F(\boldsymbol{x}_k) - F_* \leq O(\varepsilon^k)$, with $\epsilon < 1$.

Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.

Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S reconstitutes $\nabla f(\boldsymbol{x}^k)$,

Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S reconstitutes $\nabla f(\boldsymbol{x}^k)$,
- and makes the update:

$$\boldsymbol{x}^{k+1} = \operatorname{prox}_{\gamma_k g}\left(\boldsymbol{x}^k - \gamma_k \nabla f(\boldsymbol{x}^k)\right).$$

Synchronous Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S reconstitutes $\nabla f(\boldsymbol{x}^k)$,
- and makes the update:

$$\boldsymbol{x}^{k+1} = \mathrm{prox}_{\gamma_k g}\left(\boldsymbol{x}^k - \gamma_k \nabla f(\boldsymbol{x}^k)\right).$$

Synchronous Parallel Forward-Backward

Suppose that $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$, that we have a central server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S reconstitutes $\nabla f(\boldsymbol{x}^k)$,
- and makes the update:

$$\boldsymbol{x}^{k+1} = \mathrm{prox}_{\gamma_k g}\left(\boldsymbol{x}^k - \gamma_k \nabla f(\boldsymbol{x}^k)\right).$$

Synchronous algorithms are as slow as the slowest machine.

Outline

General introduction

Asynchronous Forward-Backward

Variance reduction techniques for SPPA

Conclusion

Problem setting

Consider:

$$\underset{\mathbf{x}\in\mathbf{H}}{\text{minimize}}\; F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \qquad g(\mathbf{x}) = \sum_{i=1}^{m} g_i(\mathsf{x}_i),$$

where $\mathbf{H} = \bigoplus_{i=1}^{m} \mathsf{H}_i$.

Coordinate Forward-Backward

Algorithm

Let $(\gamma_i)_{1\le i\le m} \in \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \dots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\textit{for } k = 0, 1, \dots \\
\left\lfloor
\begin{array}{l}
\textit{choose } i_k \textit{ uniformly at random in } [m] := \{1, 2, \dots, m\} \\
\textit{for } i = 1, \dots, m \\
\left\lfloor
\; x_i^{k+1} = \begin{cases} \mathrm{prox}_{\gamma_i g_i}\left(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^k)\right) & \textit{if } i = i_k \\ x_i^k & \textit{if } i \neq i_k. \end{cases}
\right.
\end{array}
\right.
\end{array}
$$

Asynchronous Forward-Backward

Server S and m machines: $M_1, M_2, \cdots, M_m$.

Asynchronous Forward-Backward

Server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.

Asynchronous Forward-Backward

Server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S receives $\nabla_{i_k} f(\boldsymbol{x}^k)$ from machine M_{i_k},

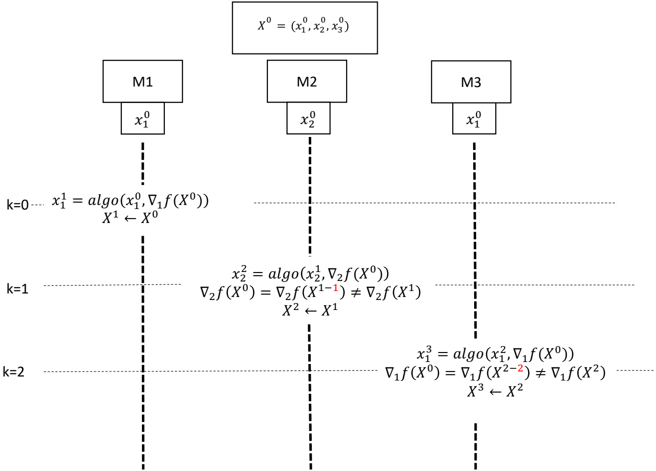
Asynchronous Forward-Backward

Server S and m machines: $M_1, M_2, \cdots, M_m$.

- For $i \in \{1, 2, \cdots, m\}$, machine M_i computes $\nabla_i f(\boldsymbol{x}^k)$.
- The server S receives $\nabla_{i_k} f(\boldsymbol{x}^k)$ from machine M_{i_k},
- and makes the update:

$$\boldsymbol{x}_{i_k}^{k+1} = \text{prox}_{\gamma_{i_k} g_{i_k}} \left(\boldsymbol{x}_{i_k}^k - \gamma_{i_k} \nabla_{i_k} f(\boldsymbol{x}^k) \right).$$

Delayed gradient in asynchronous setting



Asynchronous Forward-Backward

Algorithm

Let $(\gamma_i)_{1\le i\le m} \subset \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \ldots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\textit{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\textit{choose } i_k \textit{ uniformly at random in } [m] \\
\textit{for } i = 1, \ldots, m \\
\left\lfloor
\begin{array}{l}
x_i^{k+1} = \begin{cases} \mathrm{prox}_{\gamma_i g_i}\big(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^{k-\mathsf{d}^k})\big) & \textit{if } i = i_k \\ x_i^k & \textit{if } i \neq i_k, \end{cases}
\end{array}
\right.
\end{array}
\right.
\end{array}
$$

where $\boldsymbol{x}^{k-\mathsf{d}^k} = (x_1^{k-\mathsf{d}^k}, \ldots, x_m^{k-\mathsf{d}^k})$.

Asynchronous Forward-Backward

Algorithm

Let $(\gamma_i)_{1 \le i \le m} \subset \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \ldots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\text{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\text{choose } i_k \text{ uniformly at random in } [m] \\
\text{for } i = 1, \ldots, m \\
\left\lfloor
x_i^{k+1} = \begin{cases} \operatorname{prox}_{\gamma_i g_i}\left(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^{k-\mathsf{d}^k})\right) & \text{if } i = i_k \\ x_i^k & \text{if } i \neq i_k, \end{cases}
\right.
\end{array}
\right.
\end{array}
$$

where $\boldsymbol{x}^{k-\mathsf{d}^k} = (x_1^{k-\mathsf{d}^k}, \ldots, x_m^{k-\mathsf{d}^k})$. $\mathsf{d}^k \in \mathbb{N}$.

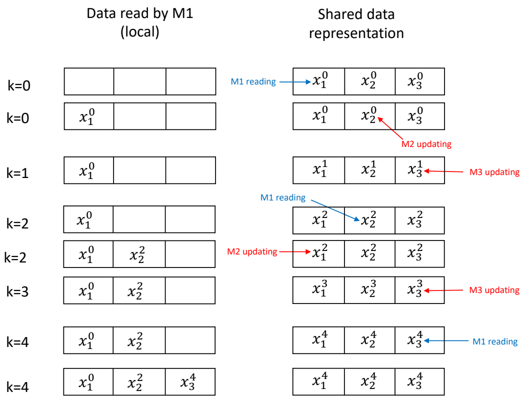
Read paradigm

- Consistent read

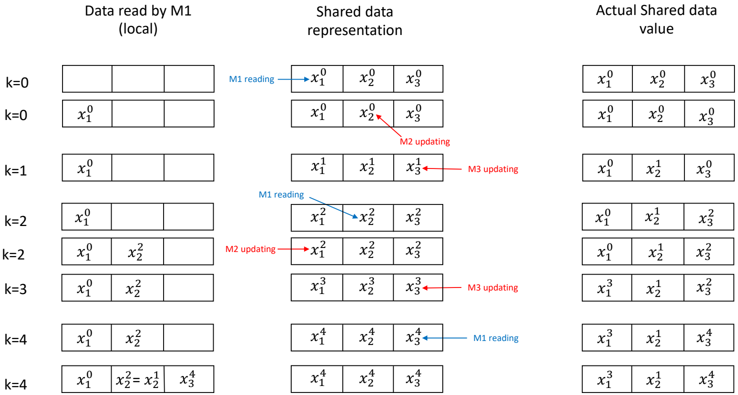
Read paradigm

- Consistent read
- Inconsistent read

Inconsistent read



Inconsistent read



Algorithm studied[2]

Algorithm

Let $(\gamma_i)_{1\leq i\leq m} \subset \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \ldots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\text{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\text{choose randomly } i_k \text{ in } [m] \text{ with probability } \mathsf{p}_{i_k} \\
\text{for } i = 1, \ldots, m \\
\left\lfloor
x_i^{k+1} = \begin{cases} \operatorname{prox}_{\gamma_i g_i}\left(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^{k-\mathbf{d}^k})\right) & \text{if } i = i_k \\ x_i^k & \text{if } i \neq i_k, \end{cases}
\right.
\end{array}
\right.
\end{array}
$$

where $\boldsymbol{x}^{k-\mathbf{d}^k} = (x_1^{k-\mathsf{d}_1^k}, \ldots, x_m^{k-\mathsf{d}_m^k})$.

[2] Traoré, Salzo, et al., “Convergence of an asynchronous block-coordinate forward-backward algorithm for convex composite optimization”.

Algorithm studied[2]

Algorithm

Let $(\gamma_i)_{1\leq i\leq m} \subset \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \dots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\text{for } k = 0, 1, \dots \\
\left\lfloor
\begin{array}{l}
\text{choose randomly } i_k \text{ in } [m] \text{ with probability } \mathsf{p}_{i_k} \\
\text{for } i = 1, \dots, m \\
\left\lfloor
x_i^{k+1} = \begin{cases} \mathrm{prox}_{\gamma_i g_i}\left(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^{k-\mathbf{d}^k})\right) & \text{if } i = i_k \\ x_i^k & \text{if } i \neq i_k, \end{cases}
\right.
\end{array}
\right.
\end{array}
$$

where $\boldsymbol{x}^{k-\mathbf{d}^k} = (x_1^{k-\mathsf{d}_1^k}, \dots, x_m^{k-\mathsf{d}_m^k})$. $\mathbf{d}^k \in \mathbb{N}^m$.

[2] Traoré, Salzo, et al., "Convergence of an asynchronous block-coordinate forward-backward algorithm for convex composite optimization".

Algorithm studied[2]

Algorithm

Let $(\gamma_i)_{1\leq i\leq m} \subset \mathbb{R}^m_{++}$ *and* $\boldsymbol{x}^0 = (x_1^0, \ldots, x_m^0) \in \mathbf{H}$. *Iterate*

$$
\begin{array}{l}
\text{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\text{choose randomly } i_k \text{ in } [m] \text{ with probability } \mathsf{p}_{i_k} \\
\text{for } i = 1, \ldots, m \\
\left\lfloor
\begin{array}{l}
x_i^{k+1} = \begin{cases} \operatorname{prox}_{\gamma_i g_i}\left(x_i^k - \gamma_i \nabla_i f(\boldsymbol{x}^{k-\mathbf{d}^k})\right) & \text{if } i = i_k \\ x_i^k & \text{if } i \neq i_k, \end{cases}
\end{array}
\right.
\end{array}
\right.
\end{array}
$$

where $\boldsymbol{x}^{k-\mathbf{d}^k} = (x_1^{k-\mathsf{d}_1^k}, \ldots, x_m^{k-\mathsf{d}_m^k})$. $\mathbf{d}^k \in \mathbb{N}^m$. $\mathsf{p}_{\max} = \max_i \mathsf{p}_i$ *and* $\mathsf{p}_{\min} = \min_i \mathsf{p}_i > 0$.

[2] Traoré, Salzo, et al., “Convergence of an asynchronous block-coordinate forward-backward algorithm for convex composite optimization”.

Assumptions

- $f\colon \mathbf{H} \to \mathbb{R}$ is convex and differentiable.
- For every $i \in \{1, \cdots, m\}$, $g_i\colon \mathsf{H}_i \to \left]-\infty, +\infty\right]$ is proper convex and lower semicontinuous.
- For all $\mathbf{x} \in \mathbf{H}$ and $i \in \{1, \cdots, m\}$, the map $\nabla_i f(\mathsf{x}_1, \dots, \mathsf{x}_{i-1}, \cdot, \mathsf{x}_{i+1}, \dots, \mathsf{x}_m)\colon \mathsf{H}_i \to \mathsf{H}_i$ is Lipschitz continuous with constant L_i. Define $L_{\max} \coloneqq \max_i L_i$ and $L_{\min} \coloneqq \min_i L_i$.
- For all $\mathbf{x} \in \mathbf{H}$ and $i \in \{1, \cdots, m\}$, the map $\nabla f(\mathsf{x}_1, \dots, \mathsf{x}_{i-1}, \cdot, \mathsf{x}_{i+1}, \dots, \mathsf{x}_m)\colon \mathsf{H}_i \to \mathbf{H}$ is Lipschitz continuous with constant $L_{\mathrm{res}} > 0$. Note that $L_{\max} \leq L_{\mathrm{res}}$.
- F attains its minimum $F_* := \min F$ on $\mathbf{H}$.

Assumptions

- We assume that the delays are deterministic and bounded by τ.
- The delay vector is independent of the coordinates.

Main result

Theorem (Convex case)

Assume that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$.

Main result

Theorem (Convex case)

Assume that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$*. Then,*

- $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ *converges weakly* P*-a.s. to* $\boldsymbol{x}^*$ *in* argmin F.

Main result

Theorem (Convex case)

Assume that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$*. Then,*

- $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ *converges weakly* P-*a.s. to* $\boldsymbol{x}^*$ *in* $\operatorname{argmin} F$.
- *And*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[F(\boldsymbol{x}^k) - F_*] \leq \frac{1}{k}\left(\frac{\operatorname{dist}^2_{\mathsf{W}}(\boldsymbol{x}^0, \operatorname{argmin} F)}{2} + C\left(F(\boldsymbol{x}^0) - F_*\right)\right),$$

where $C = O(\tau)$ *and* $\mathsf{W} = \bigoplus_{i=1}^{m} \frac{1}{\gamma_i \mathsf{p}_i}\mathsf{Id}_i$.

Linear convergence

Luo-Tseng error bound condition:

$$(\forall \mathbf{x} \in \mathsf{X}) \quad \operatorname{dist}_{\Gamma^{-1}}(\mathbf{x}, \operatorname{argmin} F) \leq C_{\mathsf{X},\Gamma^{-1}} \left\| \mathbf{x} - \operatorname{prox}_g^{\Gamma^{-1}} \left(\mathbf{x} - \nabla^{\Gamma^{-1}} f(\mathbf{x}) \right) \right\|_{\Gamma^{-1}},$$

where $\Gamma^{-1} = \bigoplus_{i=1}^{m} \frac{1}{\gamma_i} \mathsf{Id}_i$.

Linear convergence

Luo-Tseng error bound condition:

$$(\forall \mathbf{x} \in \mathsf{X}) \quad \operatorname{dist}_{\Gamma^{-1}}(\mathbf{x}, \operatorname{argmin} F) \leq C_{\mathsf{X},\Gamma^{-1}} \left\| \mathbf{x} - \operatorname{prox}_g^{\Gamma^{-1}} \left(\mathbf{x} - \nabla^{\Gamma^{-1}} f(\mathbf{x}) \right) \right\|_{\Gamma^{-1}},$$

where $\Gamma^{-1} = \bigoplus_{i=1}^{m} \frac{1}{\gamma_i} \mathsf{Id}_i$.

Equivalent to *quadratic growth*:

$$(\forall \mathbf{x} \in \mathsf{X}) \quad \frac{func(C_{\mathsf{X},\Gamma^{-1}})}{2} \operatorname{dist}^2_{\Gamma^{-1}}(\mathbf{x}, \operatorname{argmin} F) \leq F(\boldsymbol{x}_k) - F_*.$$

Linear convergence

Theorem (With error bound condition)

Suppose that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$.

Linear convergence

Theorem (With error bound condition)

Suppose that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$*. Then,*

- $(\forall k \in \mathbb{N}) \quad \mathsf{E}\big[F(\boldsymbol{x}^{k+1}) - F_*\big] \leq \left(1 - \dfrac{\mathsf{p}_{\min}}{\kappa + \theta}\right)^{\lfloor \frac{k+1}{\tau+1} \rfloor} \mathsf{E}\big[F(\boldsymbol{x}^0) - F_*\big],$

  where $\kappa \geq 1$ *and* $\theta > 0$.

Linear convergence

Theorem (With error bound condition)

Suppose that $\gamma_i < \dfrac{2}{L_i + 2\tau \frac{\mathsf{p}_{\max}}{\sqrt{\mathsf{p}_{\min}}}}$ *for all* $i \in [m]$. *Then,*

- $(\forall k \in \mathbb{N}) \quad \mathsf{E}\big[F(\boldsymbol{x}^{k+1}) - F_*\big] \leq \left(1 - \dfrac{\mathsf{p}_{\min}}{\kappa + \theta}\right)^{\lfloor \frac{k+1}{\tau+1} \rfloor} \mathsf{E}\big[F(\boldsymbol{x}^0) - F_*\big],$

  where $\kappa \geq 1$ *and* $\theta > 0$.
- $(\boldsymbol{x}^k)_{k \in \mathbb{N}}$ *converges strongly* P-*a.s. to* $\boldsymbol{x}^* \in \operatorname{argmin} F$ *and*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}\big[\|\boldsymbol{x}^k - \boldsymbol{x}^*\|_{\Gamma^{-1}}\big] = \mathcal{O}\Big(\big(1 - \mathsf{p}_{\min}/(\kappa + \theta)\big)^{\lfloor \frac{k}{\tau+1} \rfloor / 2}\Big).$$

Related works

- Liu and Wright[3]
  - constant stepsize
  - uniform probability
  - geometric (exponential) dependence of the stepsize on the delay.

[3] Liu et al., "Asynchronous stochastic coordinate descent: Parallelism and convergence properties".

Related works

- Liu and Wright[3]
  - constant stepsize
  - uniform probability
  - geometric (exponential) dependence of the stepsize on the delay.
- Salzo and Villa[4]
  - No delay consideration.

[3] Liu et al., “Asynchronous stochastic coordinate descent: Parallelism and convergence properties”.
[4] Salzo et al., “Parallel random block-coordinate forward–backward algorithm: a unified convergence analysis”.

Related works

- Liu and Wright[3]
  - constant stepsize
  - uniform probability
  - geometric (exponential) dependence of the stepsize on the delay.
- Salzo and Villa[4]
  - No delay consideration.
- Other works: Davis[5], Cannelli et al.[6]

[3] Liu et al., “Asynchronous stochastic coordinate descent: Parallelism and convergence properties”.
[4] Salzo et al., “Parallel random block-coordinate forward–backward algorithm: a unified convergence analysis”.
[5] Davis, “The asynchronous palm algorithm for nonsmooth nonconvex problems”.
[6] Cannelli et al., “Asynchronous parallel algorithms for nonconvex optimization”.

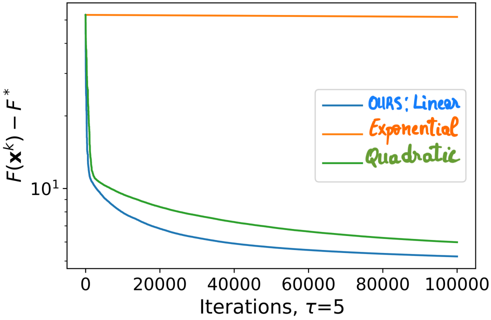
Experiments

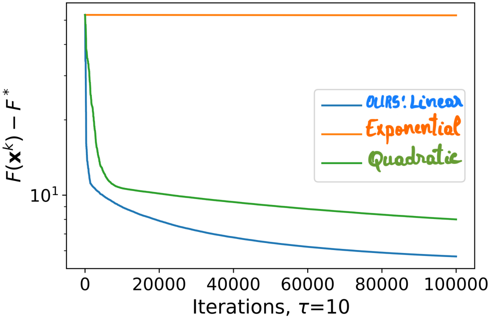
$\mathsf{A} \in \mathbb{R}^{n\times m}$ and $\mathsf{b} \in \mathbb{R}^n$,

$$\underset{\mathbf{x}\in\mathbb{R}^m}{\text{minimize}}\ \frac{1}{2}\|\mathsf{A}\mathbf{x} - \mathsf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1 \qquad (\lambda > 0)\,.$$

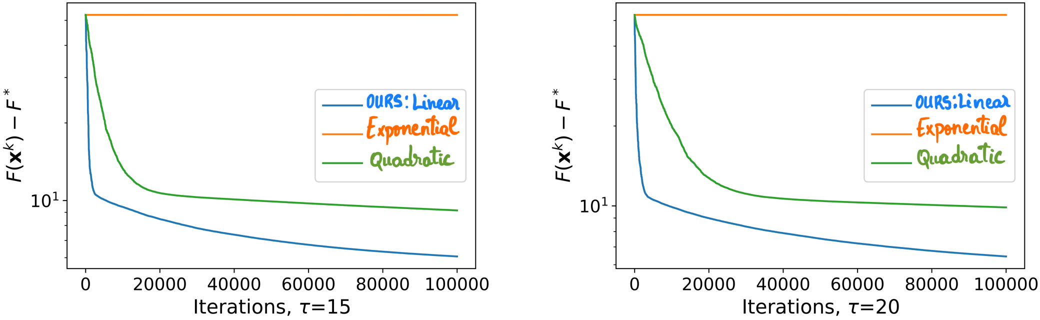
Here, $f(\mathbf{x}) = (1/2)\|\mathsf{A}\mathbf{x} - \mathsf{b}\|_2^2$ and $g_i(\mathsf{x}_i) = \lambda|\mathsf{x}_i|$.

Comparing to existing works

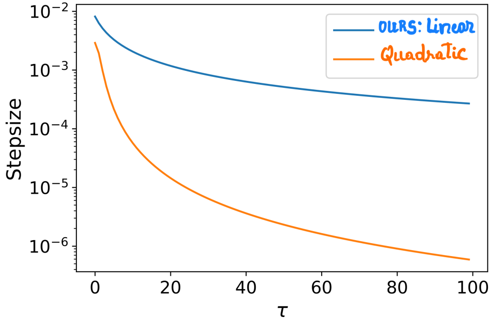


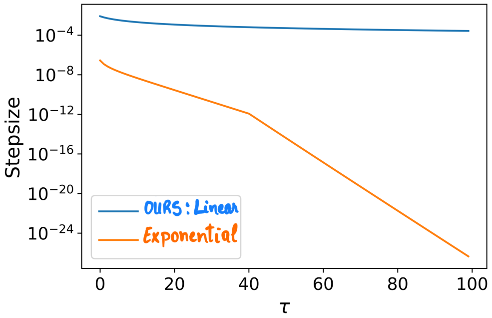


Comparing to existing works



Comparing to existing works





Remarks on the delay vector

- inconsistent read

Remarks on the delay vector

- inconsistent read
- Need τ to fix the stepsize.

Remarks on the delay vector

- inconsistent read
- Need τ to fix the stepsize.
- independence of the coordinates

Outline

General introduction

Asynchronous Forward-Backward

Variance reduction techniques for SPPA

Conclusion

Problem

Recall the first case:

$$\underset{\mathbf{x} \in \mathbf{H}}{\text{minimize}} \;\; F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}),$$

where for all $i \in [n]$, $f_i : \mathbf{H} \to \mathbb{R}$.

SGD

For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}^k)$$

SGD

For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}^k)$$

- $\gamma_k \to 0$.

SGD

For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}^k)$$

- $\gamma_k \to 0$. For instance, $\gamma_k = \dfrac{\gamma_0}{k^\beta}$ with $\beta \in [1/2, 1]$.

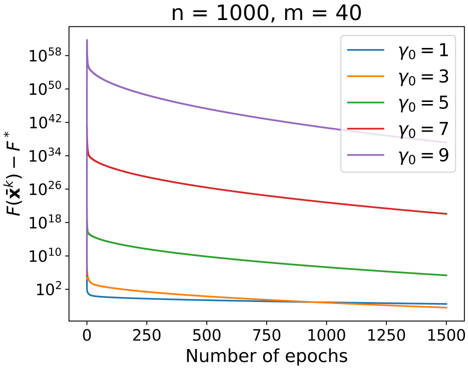
SGD

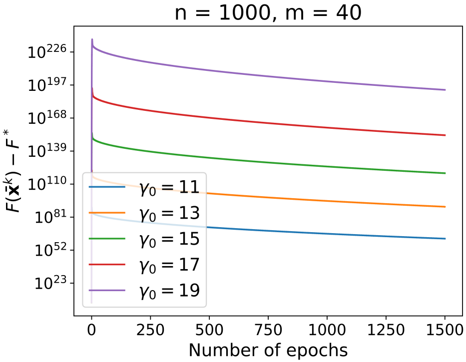
For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}^k)$$

- $\gamma_k \to 0$. For instance, $\gamma_k = \dfrac{\gamma_0}{k^\beta}$ with $\beta \in [1/2, 1]$.
- Rates in expectation: $O\left(\dfrac{1}{\sqrt{k}}\right)$ for the convex case and $O(1/k)$ for the strongly convex case.

SGD instable w.r.t. γ_0





How can we alleviate the instability with respect to γ_0?

How can we alleviate the instability with respect to γ_0?

By using, for example, stochastic proximal point algorithm (SPPA)!

SPPA

For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \mathsf{prox}_{\gamma_k f_{i_k}}(\boldsymbol{x}^k).$$

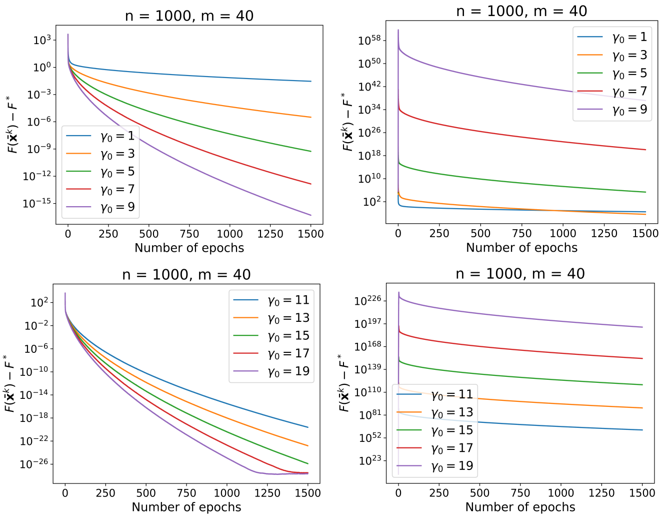
SPPA

For all $k \in \mathbb{N}$,

$$\boldsymbol{x}^{k+1} = \mathsf{prox}_{\gamma_k f_{i_k}}(\boldsymbol{x}^k).$$

Converges with same stepsize rule as SGD!!

SPPA more stable



(a) SPPA (b) SGD

SPPA more stable

From Asi and Duchi[7]:

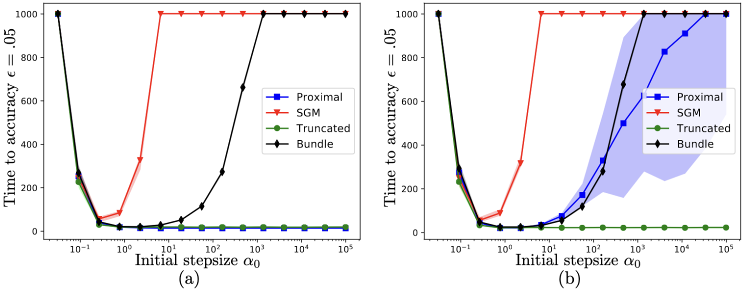


Figure 2. The number of iterations to achieve ϵ-accuracy versus initial stepsize α_0 for linear regression with $m = 1000$, $n = 40$, and condition number $\kappa(A) = 1$. (a) The noiseless setting with $\sigma = 0$. (b) Noisy setting with $\sigma = \frac{1}{2}$.

[7] Asi et al., “Stochastic (approximate) proximal point methods: convergence, optimality, and adaptivity”.

What about the rates ?

What about the rates ?

Does SPPA help recover the full GD rates ?

What about the rates ?

Does SPPA help recover the full GD rates ?

No! Same as SGD.

One problem with SPPA[8] and SGD[9] bounds

$\forall i \in [n]$ f_i convex and $L-$smooth:

$$\mathsf{E}[F(\bar{\boldsymbol{x}}^k) - F_*] \leq \frac{\mathrm{dist}(\boldsymbol{x}^0, \operatorname{argmin} F)^2}{\sum_{t=0}^{k-1} \gamma_t} + 2\sigma^2 \frac{\sum_{t=0}^{k-1} \gamma_t^2}{\sum_{t=0}^{k-1} \gamma_t},$$

[8] Traoré, Apidopoulos, et al., “Variance reduction techniques for stochastic proximal point algorithms”.
[9] Garrigos et al., “Handbook of convergence theorems for (stochastic) gradient methods”.

One problem with SPPA[8] and SGD[9] bounds

$\forall i \in [n]$ f_i convex and $L-$smooth:

$$\mathsf{E}[F(\bar{\boldsymbol{x}}^k) - F_*] \leq \frac{\mathsf{dist}(\boldsymbol{x}^0, \mathsf{argmin}\, F)^2}{\sum_{t=0}^{k-1} \gamma_t} + 2\sigma^2 \frac{\sum_{t=0}^{k-1} \gamma_t^2}{\sum_{t=0}^{k-1} \gamma_t},$$

$$\sigma^2 := \sup_{\mathbf{x}^* \in \mathsf{argmin}\, F} \mathsf{E}\|\nabla f_i(\mathbf{x}^*) - \mathsf{E}[\nabla f_i(\mathbf{x}^*)]\|^2.$$

[8] Traoré, Apidopoulos, et al., "Variance reduction techniques for stochastic proximal point algorithms".
[9] Garrigos et al., "Handbook of convergence theorems for (stochastic) gradient methods".

“Can we do better ?”, *Dr. Cris Vega, modern day philosopher*.

“Can we do better ?”, *Dr. Cris Vega, modern day philosopher*.

Yes, by using variance reduction techniques!

A way to reduce the variance of a R.V.

- X a random variable (R.V.).

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.

- Given Z, easy-to-compute $\mathrm{E}[Z]$.

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.
- Given Z, easy-to-compute $\mathsf{E}[Z]$.
- Define

$$X_Z := X - Z + \mathsf{E}[Z].$$

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.

- Given Z, easy-to-compute $\mathsf{E}[Z]$.

- Define

$$X_Z := X - Z + \mathsf{E}[Z].$$

- We want

$$\mathsf{Var}(X_Z) < \mathsf{Var}(X).$$

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.
- Given Z, easy-to-compute $\mathsf{E}[Z]$.
- Define
$$X_Z := X - Z + \mathsf{E}[Z].$$
- We want
$$\mathsf{Var}(X_Z) < \mathsf{Var}(X).$$
- We know
$$\mathsf{Var}(X_Z) = \mathsf{Var}(X) + \mathsf{Var}(Z) - 2\mathsf{Cov}(X, Z).$$

A way to reduce the variance of a R.V.

- X a random variable (R.V.).
  - Goal: reduce variance of X.

- Given Z, easy-to-compute $\mathsf{E}[Z]$.

- Define
$$X_Z := X - Z + \mathsf{E}[Z].$$

- We want
$$\mathsf{Var}(X_Z) < \mathsf{Var}(X).$$

- We know
$$\mathsf{Var}(X_Z) = \mathsf{Var}(X) + \mathsf{Var}(Z) - 2\mathsf{Cov}(X, Z).$$

- It is sufficient to have $\mathsf{Cov}(X, Z) > \frac{1}{2}\mathsf{Var}(Z)$.

General variance reduction scheme for SGD

- At each iteration k, $X = \nabla f_{i_k}(\boldsymbol{x}^k)$ and $\mathsf{E}[X|\boldsymbol{x}^k] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^k) = \nabla F(\boldsymbol{x}^k)$.

General variance reduction scheme for SGD

- At each iteration k, $X = \nabla f_{i_k}(\boldsymbol{x}^k)$ and $\mathsf{E}[X|\boldsymbol{x}^k] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^k) = \nabla F(\boldsymbol{x}^k)$.
- Goal: find a good Z and replace $X = \nabla f_{i_k}(\boldsymbol{x}^k)$ by

$$X_Z := \nabla f_{i_k}(\boldsymbol{x}^k) - Z + \mathsf{E}[Z|\boldsymbol{x}^k].$$

General variance reduction scheme for SGD

- At each iteration k, $X = \nabla f_{i_k}(\boldsymbol{x}^k)$ and $\mathsf{E}[X|\boldsymbol{x}^k] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^k) = \nabla F(\boldsymbol{x}^k)$.
- Goal: find a good Z and replace $X = \nabla f_{i_k}(\boldsymbol{x}^k)$ by

$$X_Z \coloneqq \nabla f_{i_k}(\boldsymbol{x}^k) - Z + \mathsf{E}[Z|\boldsymbol{x}^k].$$

- Depending on the choice of Z and how it is used, we get different algorithms.

At iteration k, $Z = \nabla f_{i_k}(\tilde{\boldsymbol{x}} = \boldsymbol{x}^{k-\mathsf{d}})$. Then $X_Z \coloneqq \nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\tilde{\boldsymbol{x}}) + \mathsf{E}[\nabla f_{i_k}(\tilde{\boldsymbol{x}}) \,|\, \boldsymbol{x}^k]$.

Stochastic variance reduced gradient (SVRG)[10]

At iteration k, $Z = \nabla f_{i_k}(\tilde{\boldsymbol{x}} = \boldsymbol{x}^{k-\mathsf{d}})$. Then $X_Z := \nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\tilde{\boldsymbol{x}}) + \mathsf{E}[\nabla f_{i_k}(\tilde{\boldsymbol{x}}) \,|\, \boldsymbol{x}^k]$.

Algorithm

Let $\gamma > 0$ *and set* $\tilde{\boldsymbol{x}}^0 \in \mathsf{H}$. *Then*

for $s = 0, 1, \ldots$

- $\boldsymbol{x}^0 = \tilde{\boldsymbol{x}}^s$, *compute* $\nabla F(\tilde{\boldsymbol{x}}^s)$
- *for* $k = 0, \ldots, \ell - 1$
  - *choose* i_k *uniformly at random in* $[n]$
  - $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma \left(\nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\tilde{\boldsymbol{x}}^s) + \nabla F(\tilde{\boldsymbol{x}}^s) \right)$
- *choose* ξ_s *uniformly at random in* $\{0, 1, \cdots \ell - 1\}$
- $\tilde{\boldsymbol{x}}^{s+1} = \sum_{k=0}^{\ell-1} \delta_{k,\xi_s} \boldsymbol{x}^k$,

where $\delta_{k,h}$ *is the Kronecker symbol.*

[10] Johnson et al., "Accelerating stochastic gradient descent using predictive variance reduction".

At iteration k, $Z = \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k = \boldsymbol{x}^{k-\mathsf{d}})$, with $f_{i_{k-\mathsf{d}}} = f_{i_k}$. Then

$$X_Z \coloneqq \nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) + \mathsf{E}[\nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) \,|\, \boldsymbol{x}^k].$$

Stochastic average gradient algorithm (SAGA)[11]

At iteration k, $Z = \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k = \boldsymbol{x}^{k-\mathsf{d}})$, with $f_{i_{k-\mathsf{d}}} = f_{i_k}$. Then

$$X_Z \coloneqq \nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) + \mathsf{E}[\nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) \,|\, \boldsymbol{x}^k].$$

Algorithm

Let $\gamma > 0$. *Set* $\boldsymbol{x}^0 \in \mathsf{H}$ *and,* $\forall i \in [n]$, $\boldsymbol{\phi}_i^0 = \mathbf{x}^0$. *Then*

for $k = 0, 1, \ldots$

choose i_k *uniformly at random in* $[n]$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma \left(\nabla f_{i_k}(\boldsymbol{x}^k) - \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) + \tfrac{1}{n} \textstyle\sum_{i=1}^n \nabla f_i(\boldsymbol{\phi}_i^k) \right),$$

$$\forall\, i \in [n]\colon\; \boldsymbol{\phi}_i^{k+1} = \boldsymbol{\phi}_i^k + \delta_{i,i_k}(\boldsymbol{x}^k - \boldsymbol{\phi}_i^k),$$

where $\delta_{i,j}$ *is the Kronecker symbol.*

[11] Defazio et al., “SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives”.

Variance reduction worked for SGD

Algorithm	Rates for strongly convex F	Cost of 1 Iteration
GD	$O(\varepsilon^k)$	$O(n)$
SGD	$O\left(1/k\right)$	$O(1)$
V.R.	$O(\varepsilon^k)$	$O(1)$

$O(1) =$ cost of $\nabla f_i(\boldsymbol{x})$.

Our goal: variance reduction for SPPA.

Algorithm proposed[12]

Algorithm (Generic)

Let $\gamma > 0$ *and* $\boldsymbol{x}^0 \in \mathsf{H}$. *Then*

$$\text{for } k = 0, 1, \ldots$$
$$\left\lfloor \begin{array}{l} \text{choose } i_k \text{ uniformly at random in } [n] \\ \boldsymbol{x}^{k+1} = \operatorname{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \boldsymbol{e}^k\right). \end{array} \right.$$

[12] Traoré, Apidopoulos, et al., "Variance reduction techniques for stochastic proximal point algorithms".

Algorithm proposed[12]

Algorithm (Generic)

Let $\gamma > 0$ *and* $\boldsymbol{x}^0 \in \mathsf{H}$. *Then*

$$\begin{array}{l} \textit{for } k = 0, 1, \ldots \\ \left\lfloor \begin{array}{l} \textit{choose } i_k \textit{ uniformly at random in } [n] \\ \boldsymbol{x}^{k+1} = \textit{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \boldsymbol{e}^k\right). \end{array} \right. \end{array}$$

Let $\boldsymbol{w}^k = \nabla f_{i_k}(\boldsymbol{x}^{k+1}) - \boldsymbol{e}^k$.

[12] Traoré, Apidopoulos, et al., "Variance reduction techniques for stochastic proximal point algorithms".

Algorithm proposed[12]

Algorithm (Generic)

Let $\gamma > 0$ *and* $\boldsymbol{x}^0 \in \mathsf{H}$. *Then*

$$
\begin{array}{l}
\textit{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\textit{choose } i_k \textit{ uniformly at random in } [n] \\
\boldsymbol{x}^{k+1} = \textit{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \boldsymbol{e}^k\right).
\end{array}
\right.
\end{array}
$$

Let $\boldsymbol{w}^k = \nabla f_{i_k}(\boldsymbol{x}^{k+1}) - \boldsymbol{e}^k$.

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma\left[\nabla f_{i_k}(\boldsymbol{x}^{k+1}) - \boldsymbol{e}^k\right] = \boldsymbol{x}^k - \gamma \boldsymbol{w}^k.$$

[12] Traoré, Apidopoulos, et al., “Variance reduction techniques for stochastic proximal point algorithms”.

Algorithm proposed[12]

Algorithm (Generic)

Let $\gamma > 0$ *and* $\boldsymbol{x}^0 \in \mathsf{H}$. *Then*

$$\begin{array}{l} \text{for } k = 0, 1, \ldots \\ \left\lfloor \begin{array}{l} \text{choose } i_k \text{ uniformly at random in } [n] \\ \boldsymbol{x}^{k+1} = \operatorname{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \boldsymbol{e}^k\right). \end{array} \right. \end{array}$$

Let $\boldsymbol{w}^k = \nabla f_{i_k}(\boldsymbol{x}^{k+1}) - \boldsymbol{e}^k$.

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma\left[\nabla f_{i_k}(\boldsymbol{x}^{k+1}) - \boldsymbol{e}^k\right] = \boldsymbol{x}^k - \gamma \boldsymbol{w}^k.$$

For the analysis we consider

$$\boldsymbol{v}^k := \nabla f_{i_k}(\boldsymbol{x}^k) - \boldsymbol{e}^k.$$

[12] Traoré, Apidopoulos, et al., “Variance reduction techniques for stochastic proximal point algorithms”.

Assumptions

Let $A, B, D \in \mathbb{R}_+$ and $\rho \in [0,1]$ and a real-valued random variable C such that, for every $k \in \mathbb{N}$,

1. $\mathsf{E}[\boldsymbol{e}^k \mid \mathfrak{F}_k] = 0$ a.s.,

Assumptions

Let $A, B, D \in \mathbb{R}_+$ and $\rho \in [0, 1]$ and a real-valued random variable C such that, for every $k \in \mathbb{N}$,

1. $\mathsf{E}[\boldsymbol{e}^k \mid \mathfrak{F}_k] = 0$ a.s.,
2. $\mathsf{E}\left[\|\boldsymbol{v}^k\|^2 \mid \mathfrak{F}_k\right] \leq 2A(F(\boldsymbol{x}^k) - F_*) + B\sigma_k^2 + C$ a.s.,

Assumptions

Let $A, B, D \in \mathbb{R}_+$ and $\rho \in [0,1]$ and a real-valued random variable C such that, for every $k \in \mathbb{N}$,

1. $\mathsf{E}[\boldsymbol{e}^k \mid \mathfrak{F}_k] = 0$ a.s.,
2. $\mathsf{E}\left[\|\boldsymbol{v}^k\|^2 \mid \mathfrak{F}_k\right] \leq 2A(F(\boldsymbol{x}^k) - F_*) + B\sigma_k^2 + C$ a.s.,
3. $\mathsf{E}\left[\sigma_{k+1}^2\right] \leq (1-\rho)\mathsf{E}\left[\sigma_k^2\right] + 2D\mathsf{E}[F(\boldsymbol{x}^k) - F_*]$.

Assumptions

1. $\operatorname{argmin} F \neq \varnothing$.
2. For all $i \in [n]$, f_i is convex and, moreover, L-smooth, i.e., differentiable and such that

   $$(\forall \mathbf{x}, \mathbf{y} \in \mathsf{H}) \quad \|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$$

   for some $L > 0$. As a consequence, F is convex and L-smooth.
3. F satisfies the PL condition with constant $\mu > 0$, i.e.,

   $$(\forall \mathbf{x} \in \mathsf{H}) \quad F(\mathbf{x}) - F_* \leq \frac{1}{2\mu}\|\nabla F(\mathbf{x})\|^2.$$

Common result

Proposition

Let $M > 0$. Then, for all $k \in \mathbb{N}$,

$$\begin{aligned}\mathsf{E}[\operatorname{dist}(\boldsymbol{x}^{k+1}, \operatorname{argmin} F)^2] + \gamma^2 M \mathsf{E}[\sigma_{k+1}^2] \leq \mathsf{E}[\operatorname{dist}(\boldsymbol{x}^{k}, \operatorname{argmin} F)^2] &+ \gamma^2 [M + B - \rho M]\, \mathsf{E}[\sigma_k^2] \\ &- 2\gamma [1 - \gamma(A + MD)]\, \mathsf{E}[F(\boldsymbol{x}^k) - F_*] \\ &+ \gamma^2 \mathsf{E}[C].\end{aligned}$$

Convex F

Theorem

Let $M > 0$ and $\gamma > 0$ be such that $M \geq B/\rho$ and $\gamma < 1/(A + MD)$. Then,

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[F(\bar{\boldsymbol{x}}^k) - F_*] \leq \frac{\operatorname{dist}(\boldsymbol{x}^0, \operatorname{argmin} F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]}{2\gamma k\,[1 - \gamma(A + MD)]},$$

with $\bar{\boldsymbol{x}}^k = \frac{1}{k}\sum_{t=0}^{k-1} \boldsymbol{x}^t$.

F satisfying PL

Theorem

Let M *be such that* $M > B/\rho$ *and* $\gamma > 0$ *such that* $\gamma < 1/(A + MD)$.
Set

$$q := \max\left\{1 - \gamma\mu\left(1 - \gamma(A + MD)\right), 1 + \frac{B}{M} - \rho\right\}.$$

Then $q \in]0, 1[$ *and*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[\mathrm{dist}(\boldsymbol{x}^k, \mathrm{argmin}\, F)^2] \leq q^k \left(\mathrm{dist}(\boldsymbol{x}^0, \mathrm{argmin}\, F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right),$$

$$\mathsf{E}[F(\boldsymbol{x}^k) - F_*] \leq \frac{q^k L}{2} \left(\mathrm{dist}(\boldsymbol{x}^0, \mathrm{argmin}\, F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right).$$

Algorithm (SVRP)

Let $\gamma > 0$ *and* $\tilde{\boldsymbol{x}}^0 \in$ H. *Then*

for $s = 0, 1, \ldots$

$\boldsymbol{x}^0 = \tilde{\boldsymbol{x}}^s$, compute $\nabla F(\tilde{\boldsymbol{x}}^s)$

for $k = 0, \ldots, \ell - 1$

choose i_k *uniformly at random in* $[n]$

$\boldsymbol{x}^{k+1} = \mathsf{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \nabla f_{i_k}(\tilde{\boldsymbol{x}}^s) - \gamma \nabla F(\tilde{\boldsymbol{x}}^s)\right)$

choose ξ_s *uniformly at random in* $\{0, 1, \cdots \ell - 1\}$

$\tilde{\boldsymbol{x}}^{s+1} = \sum_{k=0}^{\ell-1} \delta_{k,\xi_s} \boldsymbol{x}^k,$

where $\delta_{k,h}$ *is the Kronecker symbol.*

SVRP results

Theorem (PL case)

Suppose that

$$0 < \gamma < \frac{1}{2(2L-\mu)} \quad \textit{and} \quad \ell > \frac{1}{\mu\gamma(1-2\gamma(2L-\mu))}.$$

Then

$$(\forall s \in \mathbb{N}) \quad \mathsf{E}\left[F\left(\tilde{\boldsymbol{x}}^{s+1}\right) - F_*\right] \leq q^s \left(F(\boldsymbol{x}^0) - F_*\right),$$

with $q := \left(\frac{1}{\mu\gamma(1-2L\gamma)\ell} + \frac{2\gamma(L-\mu)}{1-2L\gamma}\right) < 1.$

L-SVRP

Algorithm (L-SVRP)

Let $\gamma > 0$ *and set* $\boldsymbol{x}^0 = \boldsymbol{u}^0 \in \mathsf{H}$. *Then*

$$
\begin{array}{l}
\text{for } k = 0, 1, \ldots \\
\left\lfloor \begin{array}{l}
\text{choose } i_k \text{ uniformly at random in } [n] \\
\boldsymbol{x}^{k+1} = \mathrm{prox}_{\gamma f_{i_k}}\left(\boldsymbol{x}^k + \gamma \nabla f_{i_k}(\boldsymbol{u}^k) - \gamma \nabla F(\boldsymbol{u}^k)\right) \\
\varepsilon^k \text{ Bernoulli r.v. with } \mathsf{P}(\varepsilon^k = 1) = p \in]0,1] \\
\boldsymbol{u}^{k+1} = (1 - \varepsilon^k)\boldsymbol{u}^k + \varepsilon^k \boldsymbol{x}^k,
\end{array} \right.
\end{array}
$$

L-SVRP results

Corollary (Convex case)

Let $M \geq \frac{2}{p}$ *and* $\gamma < \frac{1}{L(2+pM)}$. *Then*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[F(\bar{\boldsymbol{x}}^k) - F_*] \leq \frac{\operatorname{dist}(\boldsymbol{x}^0, \operatorname{argmin} F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]}{2\gamma k \left[1 - \gamma L(2 + pM)\right]},$$

with $\bar{\boldsymbol{x}}^k = \frac{1}{k} \sum_{t=0}^{k-1} \boldsymbol{x}^t$.

L-SVRP results

Corollary (PL case)

Let $M > 2/p$ *and* $\gamma < \frac{1}{L(2+pM)}$. *Then*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[\mathsf{dist}(\boldsymbol{x}^k, \mathsf{argmin}\, F)]^2 \leq q^k \left(\mathsf{dist}(\boldsymbol{x}^0, \mathsf{argmin}\, F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right),$$

$$\mathsf{E}[F(\boldsymbol{x}^k) - F_*] \leq \frac{q^k L}{2} \left(\mathsf{dist}(\boldsymbol{x}^0, \mathsf{argmin}\, F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right),$$

with $0 < q < 1$.

SAPA

Algorithm (**SAPA**)

Let $\gamma > 0$. *Set* $\boldsymbol{x}^0 \in \mathsf{H}$ *and,* $\forall i \in [n]$, $\boldsymbol{\phi}_i^0 = \mathbf{x}^0$. *Then*

$$
\begin{array}{l}
\textit{for } k = 0, 1, \ldots \\
\left\lfloor
\begin{array}{l}
\textit{choose } i_k \textit{ uniformly at random in } [n] \\
\boldsymbol{x}^{k+1} = \mathrm{prox}_{\gamma f_{i_k}} \left(\boldsymbol{x}^k + \gamma \nabla f_{i_k}(\boldsymbol{\phi}_{i_k}^k) - \frac{\gamma}{n} \sum_{i=1}^n \nabla f_i(\boldsymbol{\phi}_i^k) \right) \\
\forall\, i \in [n]\colon\ \boldsymbol{\phi}_i^{k+1} = \boldsymbol{\phi}_i^k + \delta_{i,i_k}(\boldsymbol{x}^k - \boldsymbol{\phi}_i^k),
\end{array}
\right.
\end{array}
$$

where $\delta_{i,j}$ *is the Kronecker symbol.*

SAPA results

Corollary (Convex case)

Let $M \geq 2n$ *and* $\gamma < \frac{1}{L(2+M/n)}$. *Then*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[F(\bar{\boldsymbol{x}}^k) - F_*] \leq \frac{\mathsf{dist}(\boldsymbol{x}^0, \mathsf{argmin}\, F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]}{2\gamma k \left[1 - \gamma L(2 + M/n)\right]},$$

with $\bar{\boldsymbol{x}}^k = \frac{1}{k}\sum_{t=0}^{k-1} \boldsymbol{x}^t$.

SAPA results

Corollary (PL case)

Let $M > 2n$ *and* $\gamma < \frac{1}{L(2+M/n)}$. *Then*

$$(\forall k \in \mathbb{N}) \quad \mathsf{E}[\operatorname{dist}(\boldsymbol{x}^k, \operatorname{argmin} F)]^2 \leq q^k \left(\operatorname{dist}(\boldsymbol{x}^0, \operatorname{argmin} F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right),$$

$$\mathsf{E}[F(\boldsymbol{x}^k) - F_*] \leq \frac{q^k L}{2} \left(\operatorname{dist}(\boldsymbol{x}^0, \operatorname{argmin} F)^2 + \gamma^2 M \mathsf{E}[\sigma_0^2]\right),$$

with $0 < q < 1$.

Related works

	Algorithm	Smooth + convex	Smooth + SC	Smooth + PL	Non-smooth + SC
Defazio[13]	Point-Saga	NA	$O(\varepsilon^k)$	NA	$O(1/k)$
Khaled et al.[14]	L-SVRP	NA	$O(\varepsilon^k)$	NA	NA
Milzarek et al.[15]	SNSPP	NA	$O(\varepsilon^k)$	NA	NA
Traoré et al.[16]	Unified	$O(1/k)$ (not for SVRP)	$O(\varepsilon^k)$	$O(\varepsilon^k)$	NA

Table: Comparison to related works.

[13] Defazio, "A simple practical accelerated method for finite sums".
[14] Khaled et al., "Faster federated optimization under second-order similarity".
[15] Milzarek et al., "A semismooth Newton stochastic proximal point algorithm with variance reduction".
[16] Traoré, Apidopoulos, et al., "Variance reduction techniques for stochastic proximal point algorithms".

Experiments

Ordinary least squares (OLS):

$$\underset{\mathbf{x}\in\mathbb{R}^m}{\text{minimize}} \;\; F(\mathbf{x}) = \frac{1}{2n}\|A\mathbf{x} - \boldsymbol{b}\|^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\langle \mathbf{a}_i, \mathbf{x}\rangle - b_i\right)^2,$$

where $\mathbf{a}_i$ is the i^{th} row of the matrix $A \in \mathbb{R}^{n\times m}$ and $b_i \in \mathbb{R}$ for all $i \in [n]$

SAPA/SVRP/SPPA

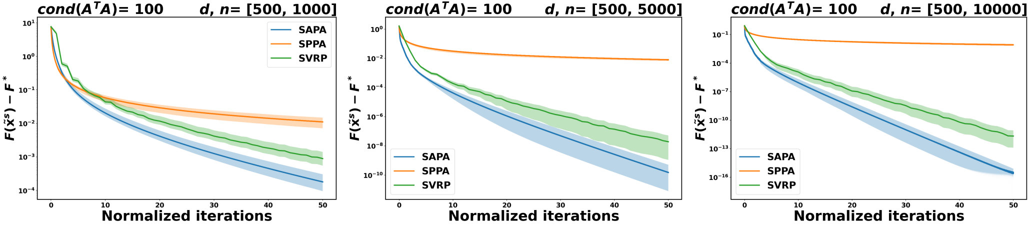


Figure: SAPA (blue) and SVRP (green) compared to SPPA (orange).

SAGA/SAPA

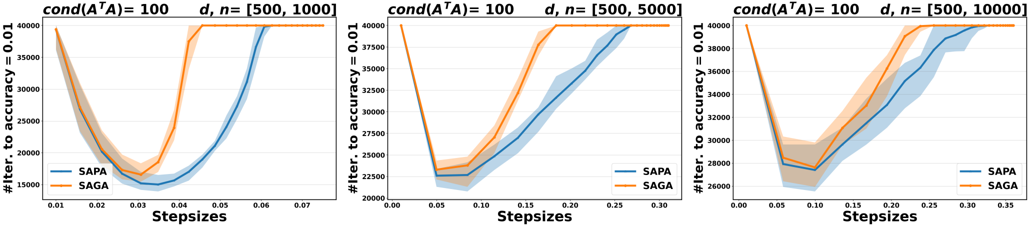


Figure: SAPA (in blue) compared to SAGA (in orange).

SVRG/SVRP

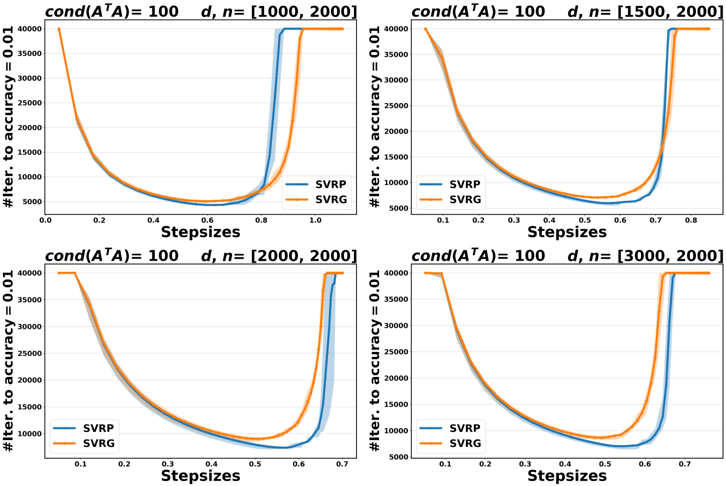


Figure: SVRP (in blue) compared SVRG (in orange).

Remarks on the results

- We only have results for the smooth case.

Remarks on the results

- We only have results for the smooth case.
- For SVRP, no results for the convex case.

Remarks on the results

- We only have results for the smooth case.
- For SVRP, no results for the convex case.
- We have derived results for SPPA from the generic algorithm.

Outline

General introduction

Asynchronous Forward-Backward

Variance reduction techniques for SPPA

Conclusion

Conclusion

Asynchronous Forward-Backward

Variance reduction for SPPA

Conclusion

Asynchronous Forward-Backward

Summary

- considered with abstract probability and coordinate-wise adaptive stepsize
- provided convergence of the iterates
- provided standard convergence rates for convex and error bound cases
- results depend linearly on τ

Variance reduction for SPPA

Summary

- made a unified study
- proved standard rates for convex and PL cases
- recovered proximal version of standard gradient variance reduced algorithms
- empirical comparison

Conclusion

Asynchronous Forward-Backward

Summary

- considered with abstract probability and coordinate-wise adaptive stepsize
- provided convergence of the iterates
- provided standard convergence rates for convex and error bound cases
- results depend linearly on τ

Future directions

- results for delay-wise adaptive stepsize
- considered coordinates dependent delay
- prove the tightest delay dependence

Variance reduction for SPPA

Summary

- made a unified study
- proved standard rates for convex and PL cases
- recovered proximal version of standard gradient variance reduced algorithms
- empirical comparison

Future directions

- convex case for SVRP
- non-smooth analysis
- Bregman and/or zeroth-order versions

Conclusion

Asynchronous Forward-Backward

Summary

- considered with abstract probability and coordinate-wise adaptive stepsize
- provided convergence of the iterates
- provided standard convergence rates for convex and error bound cases
- results depend linearly on τ

Future directions

- results for delay-wise adaptive stepsize
- considered coordinates dependent delay
- prove the tightest delay dependence

Variance reduction for SPPA

Summary

- made a unified study
- proved standard rates for convex and PL cases
- recovered proximal version of standard gradient variance reduced algorithms
- empirical comparison

Future directions

- convex case for SVRP
- non-smooth analysis
- Bregman and/or zeroth-order versions

Conclusion

Asynchronous Forward-Backward

Summary

- considered with abstract probability and coordinate-wise adaptive stepsize
- provided convergence of the iterates
- provided standard convergence rates for convex and error bound cases
- results depend linearly on τ

Future directions

- results for delay-wise adaptive stepsize
- considered coordinates dependent delay
- prove the tightest delay dependence

Variance reduction for SPPA

Summary

- made a unified study
- proved standard rates for convex and PL cases
- recovered proximal version of standard gradient variance reduced algorithms
- empirical comparison

Future directions

- convex case for SVRP
- non-smooth analysis
- Bregman and/or zeroth-order versions

Thank you for your attention!!