Minimizing Finite Sums with the Stochastic Average Gradient Algorithm

Mark Schmidt

Joint work with Nicolas Le Roux and Francis Bach

University of British Columbia

BIRS Workshop on Sparse Representations, Numerical Linear Algebra, and Optimization

Context: Machine Learning for "Big Data"

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} g(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

Context: Machine Learning for "Big Data"

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} g(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

- Applications to any data-oriented field:
  - Vision, bioinformatics, speech, natural language, web.
- We are interested in cases where N is very large.

Context: Machine Learning for "Big Data"

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} g(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

- Applications to any data-oriented field:
  - Vision, bioinformatics, speech, natural language, web.
- We are interested in cases where N is very large.
- We will focus on strongly-convex functions g.
- Simplest example is ℓ_2-regularized least-squares,

$$f_i(x) := (a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2.$$

Context: Machine Learning for "Big Data"

- We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^P} g(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

- Applications to any data-oriented field:
  - Vision, bioinformatics, speech, natural language, web.
- We are interested in cases where N is very large.
- We will focus on strongly-convex functions g.
- Simplest example is ℓ_2-regularized least-squares,

$$f_i(x) := (a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2.$$

- Other examples include any ℓ_2-regularized convex loss:
  - logistic regression, Huber regression, smooth SVMs, CRFs, etc.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $g(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $g(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

$$x_{t+1} = x_t - \alpha_t g'(x_t) = x_t - \frac{\alpha_t}{N}\sum_{i=1}^{N} f_i'(x_t).$$

  - Linear convergence rate: $O(\rho^t)$.
  - Iteration cost is linear in N.
  - Fancier methods exist, but still in $O(N)$.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $g(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

$$x_{t+1} = x_t - \alpha_t g'(x_t) = x_t - \frac{\alpha_t}{N}\sum_{i=1}^{N} f_i'(x_t).$$

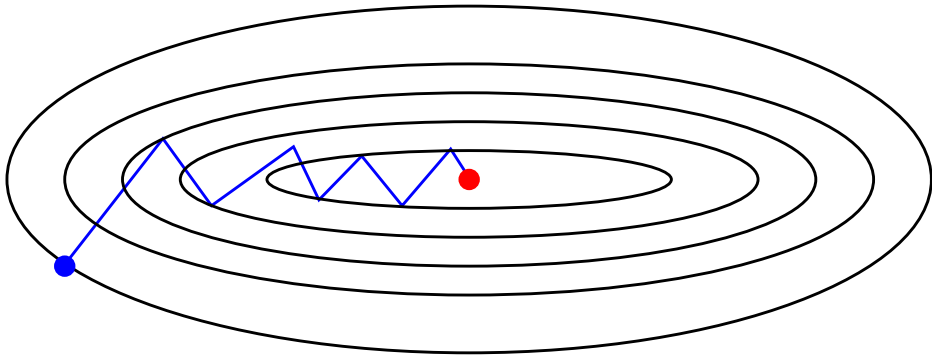
  - Linear convergence rate: $O(\rho^t)$.
  - Iteration cost is linear in N.
  - Fancier methods exist, but still in $O(N)$.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of $i(t)$ from $\{1, 2, \ldots, N\}$,

$$x_{t+1} = x_t - \alpha_t f_{i(t)}'(x_t).$$

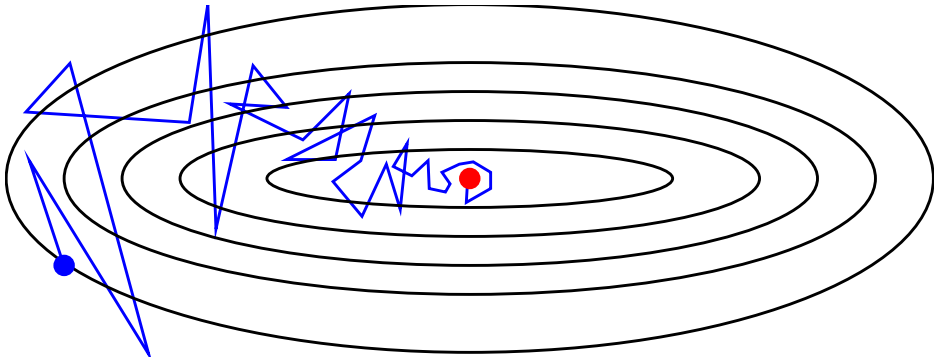
  - Iteration cost is independent of N.
  - Sublinear convergence rate: $O(1/t)$.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $g(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.
- Deterministic gradient method [Cauchy, 1847]:

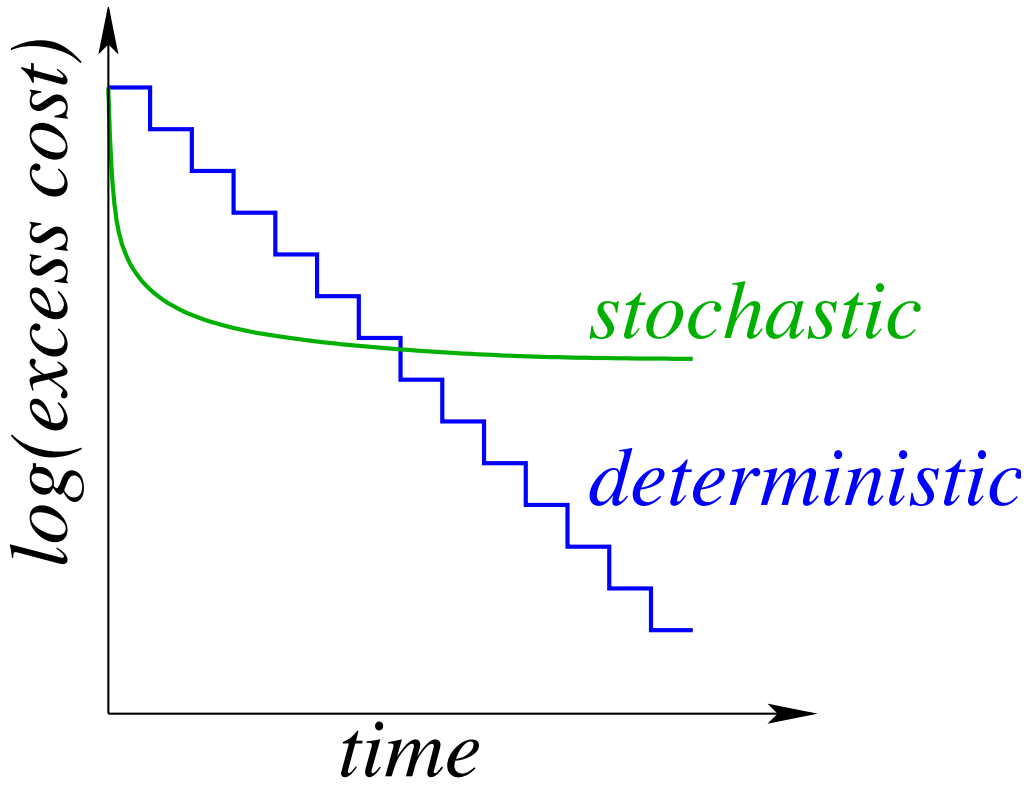


- Stochastic gradient method [Robbins & Monro, 1951]:



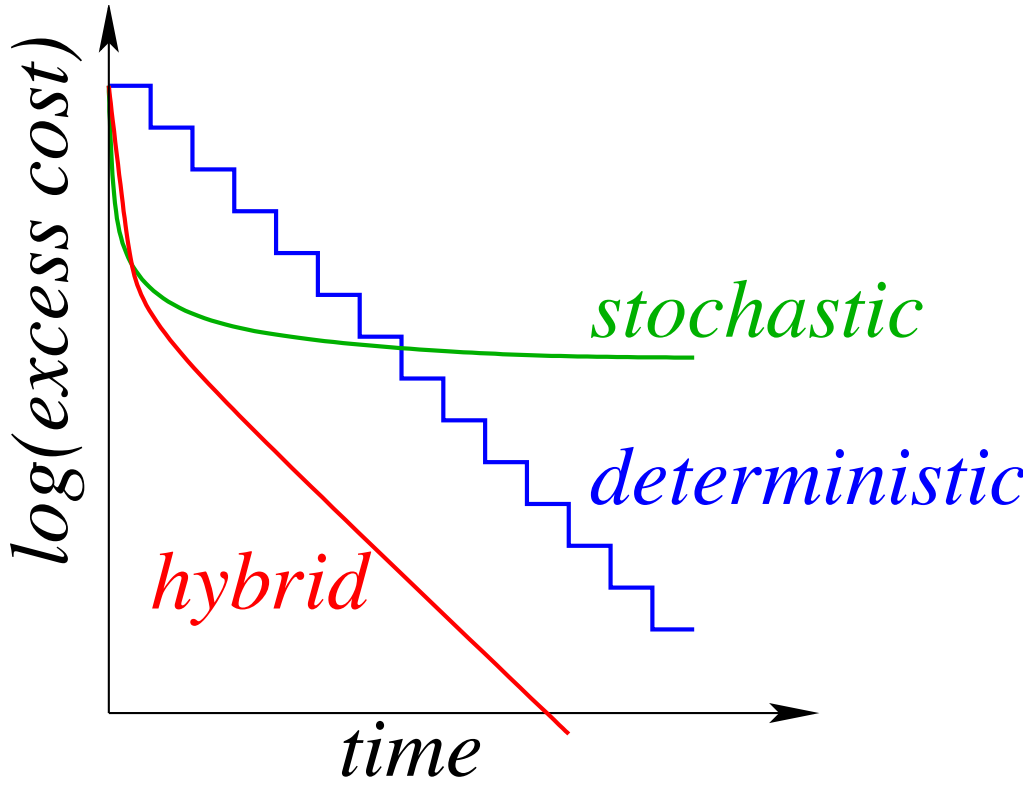
Motivation for New Methods

- FG method has $O(N)$ cost with $O(\rho^t)$ rate.
- SG method has $O(1)$ cost with $O(1/t)$ rate.



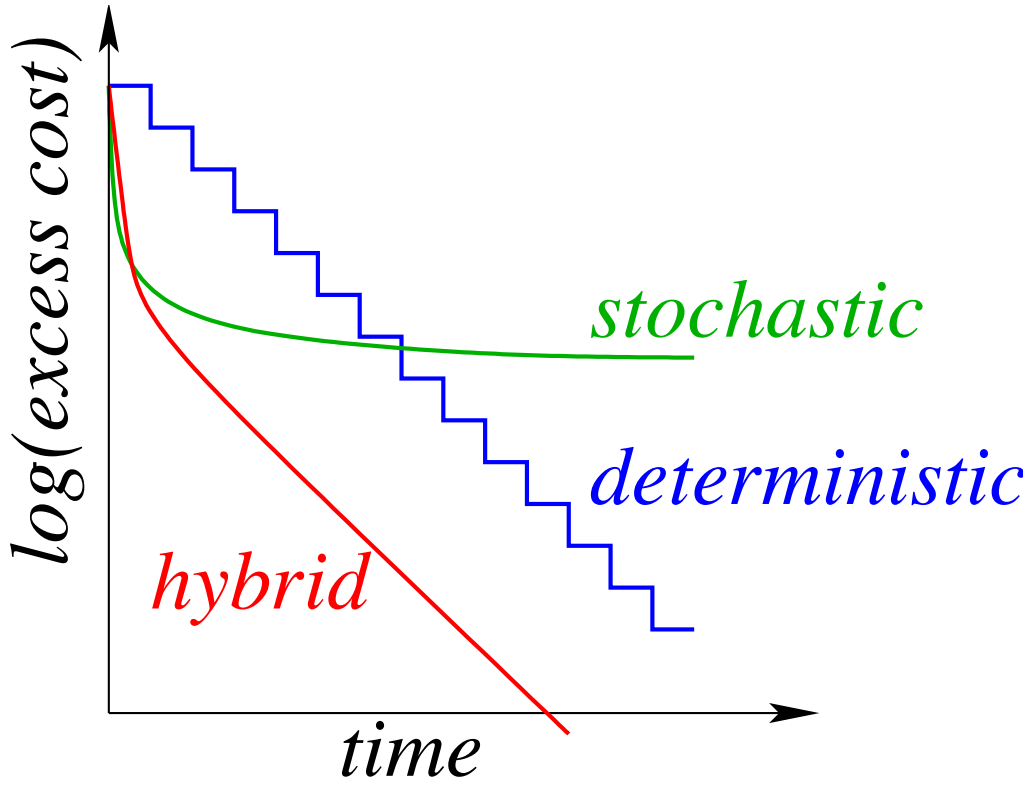
Motivation for New Methods

- FG method has $O(N)$ cost with $O(\rho^t)$ rate.
- SG method has $O(1)$ cost with $O(1/t)$ rate.



Motivation for New Methods

- FG method has $O(N)$ cost with $O(\rho^t)$ rate.
- SG method has $O(1)$ cost with $O(1/t)$ rate.



- Goal is $O(1)$ cost with $O(\rho^t)$ rate.

Prior Work on Speeding up SG Methods

A variety of methods have been proposed to speed up SG methods:

- **Step-size strategies, momentum, gradient/iterate averaging**

  [Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)]

- **Stochastic version of accelerated and Newton-like methods**

  [Bordes et al. (2009), Sunehag et al. (2009), Ghadimi and Lan (2010), Martens (2010), Xiao (2010), Duchi et al. (2011)]

Prior Work on Speeding up SG Methods

A variety of methods have been proposed to speed up SG methods:

- **Step-size strategies, momentum, gradient/iterate averaging**

  [Polyak & Juditsky (1992), Tseng (1998), Kushner & Yin (2003) Nesterov (2009), Xiao (2010), Hazan & Kale (2011), Rakhlin et al. (2012)]

- **Stochastic version of accelerated and Newton-like methods**

  [Bordes et al. (2009), Sunehag et al. (2009), Ghadimi and Lan (2010), Martens (2010), Xiao (2010), Duchi et al. (2011)]

- None of these methods improve on the $O(1/t)$ rate

Prior Work on Speeding up SG Methods

Existing linear convergence results:

- **Constant step-size SG, accelerated SG**

  [Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)]

  - Linear convergence up to a fixed tolerance: $O(\rho^t) + O(\alpha)$.

Prior Work on Speeding up SG Methods

Existing linear convergence results:

- **Constant step-size SG, accelerated SG**

  [Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)]

  - Linear convergence up to a fixed tolerance: $O(\rho^t) + O(\alpha)$.

- **Hybrid methods, incremental average gradient**

  [Bertsekas (1997), Blatt et al. (2007), Friedlander and Schmidt (2012)]

  - Linear rate but iterations make full passes through the data

Prior Work on Speeding up SG Methods

Existing linear convergence results:

- **Constant step-size SG, accelerated SG**

  [Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)]

  - Linear convergence up to a fixed tolerance: $O(\rho^t) + O(\alpha)$.

- **Hybrid methods, incremental average gradient**

  [Bertsekas (1997), Blatt et al. (2007), Friedlander and Schmidt (2012)]

  - Linear rate but iterations make full passes through the data

- **Special Problems Classes**

  [Collins et al. (2008), Strohmer & Vershynin (2009), Schmidt and Le Roux (2012), Shalev-Shwartz and Zhang (2013)]

  - Linear rate but limited choice for the f_i's

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**
  - YES!

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f_i'(x^t)$$

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = f_i'(x^t)$ from the last iteration t where i was selected.

Stochastic Average Gradient

- Assume only that:
  - f_i is convex, f_i' is $L-$continuous, g is μ-strongly convex.
- **Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i(t)$ from $\{1, 2, \dots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = f_i'(x^t)$ from the last iteration t where i was selected.
  - Assumes that gradients of other examples don't change.
  - This assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.
  - Stochastic variant of increment aggregated gradient (IAG). [Blatt et al. 2007]

Convergence Rate of SAG: Attempt 1

Proposition 1. *With* $\alpha_t = \frac{1}{2nL}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{\mu}{8LN}\right)^t C.$$

Convergence Rate of SAG: Attempt 1

Proposition 1. *With* $\alpha_t = \frac{1}{2nL}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{\mu}{8LN}\right)^t C.$$

- **Convergence rate of** $O(\rho^t)$ **with cost of** $O(1)$.
- Mission accomplished?!?

Convergence Rate of SAG: Attempt 1

Proposition 1. *With* $\alpha_t = \frac{1}{2nL}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{\mu}{8LN}\right)^t C.$$

- **Convergence rate of** $O(\rho^t)$ **with cost of** $O(1)$.
- Mission accomplished?!?
- This rate is very slow: performance similar to cyclic method.

Convergence Rate of SAG: Attempt 2

Proposition 2. *With* $\alpha_t \in [\frac{1}{2n\mu}, \frac{1}{16L}]$ *and* $N \geq 8\frac{L}{\mu}$*, the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{1}{8N}\right)^t C.$$

Convergence Rate of SAG: Attempt 2

Proposition 2. *With* $\alpha_t \in [\frac{1}{2n\mu}, \frac{1}{16L}]$ *and* $N \geq 8\frac{L}{\mu}$*, the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{1}{8N}\right)^t C.$$

- Much bigger step-sizes: $\mu << L$ (this does not work for cyclic)

Convergence Rate of SAG: Attempt 2

Proposition 2. *With* $\alpha_t \in [\frac{1}{2n\mu}, \frac{1}{16L}]$ *and* $N \geq 8\frac{L}{\mu}$*, the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \frac{1}{8N}\right)^t C.$$

- Much bigger step-sizes: $\mu << L$ (this does not work for cyclic)
- Gives constant non-trivial reduction per pass:

$$\left(1 - \frac{1}{8N}\right)^N \leq \exp\left(-\frac{1}{8}\right) = 0.8825.$$

- $N \geq O(\frac{L}{\mu})$ has been called 'big data' condition.

Convergence Rate of SAG: Attempt 3

Theorem. *With* $\alpha_t = \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C.$$

Convergence Rate of SAG: Attempt 3

Theorem. *With* $\alpha_t = \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C.$$

- This rate is "very fast":
  - Well-conditioned problems: constant non-trivial reduction per pass.

Convergence Rate of SAG: Attempt 3

Theorem. *With* $\alpha_t = \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(x^t) - g(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C.$$

- This rate is "very fast":
  - Well-conditioned problems: constant non-trivial reduction per pass.
  - Badly-conditioned problems: almost same as deterministic method:

$$g(x^t) - g(x^*) \leqslant \left(1 - \frac{\mu}{L}\right)^{2t} C,$$

with $\alpha_t = \frac{1}{L}$, but SAG is N times faster.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$.
  - SAG: $O(\max\{N, \frac{L}{\mu}\}\log(1/\epsilon))$.

Proof Technique: Lyapunov Function

- We define a Lyapunov function of the form

$$\mathcal{L}(\theta^t) = 2h[g(x^t + de^\top y^t) - g(x^*)] + (\theta^t - \theta^*)^\top \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} (\theta^t - \theta^*),$$

with

$$\theta^t = \begin{bmatrix} y_1^t \\ \vdots \\ y_t^N \\ x^t \end{bmatrix}, \quad \theta^* = \begin{bmatrix} f_i'(x^*) \\ \vdots \\ f_N'(x^*) \\ x^* \end{bmatrix}, \quad e = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, \quad \begin{aligned} A &= a_1 ee^\top + a_2 I, \\ B &= be, \\ C &= cI. \end{aligned}$$

Proof Technique: Lyapunov Function

- We define a Lyapunov function of the form

$$\mathcal{L}(\theta^t) = 2h[g(x^t + de^\top y^t) - g(x^*)] + (\theta^t - \theta^*)^\top \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} (\theta^t - \theta^*),$$

with

$$\theta^t = \begin{bmatrix} y_1^t \\ \vdots \\ y_t^N \\ x^t \end{bmatrix}, \quad \theta^* = \begin{bmatrix} f_i'(x^*) \\ \vdots \\ f_N'(x^*) \\ x^* \end{bmatrix}, \quad e = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, \quad \begin{aligned} A &= a_1 ee^\top + a_2 I, \\ B &= be, \\ C &= cI. \end{aligned}$$

- Proof involves finding $\{\alpha, a_1, a_2, b, c, d, h, \delta, \gamma\}$ such that

$$\mathbb{E}(\mathcal{L}(\theta^t)|\mathcal{F}_{t-1}) \leqslant (1-\delta)\mathcal{L}(\theta^{t-1}), \quad \mathcal{L}(\theta^t) \geqslant \gamma[g(x^t) - g(x^*)].$$

- Apply recursively and initial Lyapunov function gives constant.

Constants in Convergence Rate

- What are the constants?

Constants in Convergence Rate

- What are the constants?
  - If we initialize with $y_i^0 = 0$ we have

$$C = [g(x^0) - g(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

Constants in Convergence Rate

- What are the constants?
  - If we initialize with $y_i^0 = 0$ we have

$$C = [g(x^0) - g(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

  - If we initialize with $y_i^0 = f_i'(x^0) - g'(x^0)$ we have

$$C = \frac{3}{2}[g(x^0) - g(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2.$$

Constants in Convergence Rate

- What are the constants?
  - If we initialize with $y_i^0 = 0$ we have

$$C = [g(x^0) - g(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

  - If we initialize with $y_i^0 = f_i'(x^0) - g'(x^0)$ we have

$$C = \frac{3}{2}[g(x^0) - g(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2.$$

  - If we initialize with N stochastic gradient iterations,

$$[g(x^0) - g(x^*)] = O(1/N), \quad \|x^0 - x^*\|^2 = O(1/N).$$

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

Theorem. *With* $\alpha_t \leqslant \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(\bar{x}^t) - g(x^*)] = O(1/t)$$

- **Faster than SG lower bound of** $O(1/\sqrt{t})$.

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

Theorem. *With* $\alpha_t \leqslant \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(\bar{x}^t) - g(x^*)] = O(1/t)$$

- **Faster than SG lower bound of** $O(1/\sqrt{t})$.
- Same algorithm and step-size as strongly-convex case:
  - Algorithm is adaptive to strong-convexity.
  - Faster convergence rate if μ is locally bigger around x^*.

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

Theorem. *With* $\alpha_t \leqslant \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(\bar{x}^t) - g(x^*)] = O(1/t)$$

- **Faster than SG lower bound of** $O(1/\sqrt{t})$.
- Same algorithm and step-size as strongly-convex case:
  - Algorithm is adaptive to strong-convexity.
  - Faster convergence rate if μ is locally bigger around x^*.
- Same algorithm could be used in non-convex case.

Convergence Rate in Convex Case

Assume only that:

- f_i is convex, f_i' is $L-$continuous, some x^* exists.

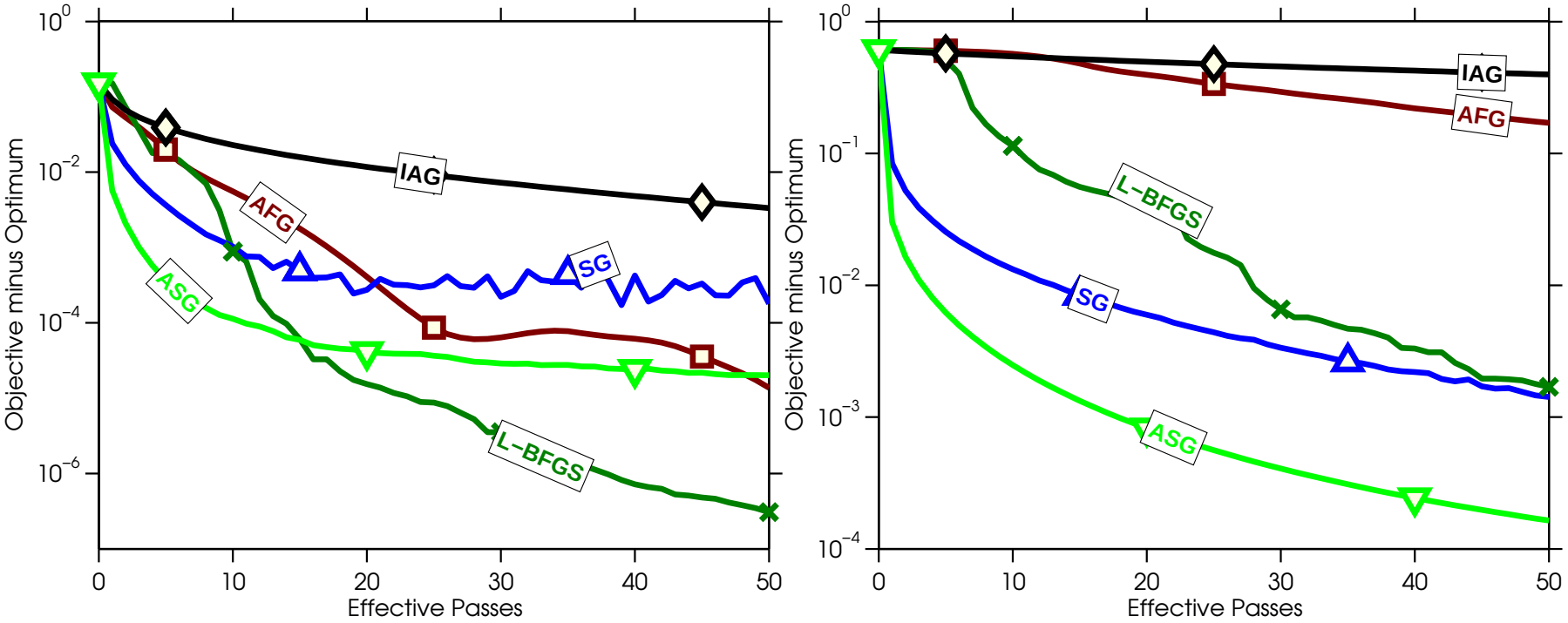
Theorem. *With* $\alpha_t \leqslant \frac{1}{16L}$ *the SAG iterations satisfy*

$$\mathbb{E}[g(\bar{x}^t) - g(x^*)] = O(1/t)$$

- **Faster than SG lower bound of** $O(1/\sqrt{t})$.
- Same algorithm and step-size as strongly-convex case:
  - Algorithm is adaptive to strong-convexity.
  - Faster convergence rate if μ is locally bigger around x^*.
- Same algorithm could be used in non-convex case.
- Contrast with stochastic dual coordinate ascent:
  - Requires explicit strongly-convex regularizer.
  - Not adaptive to μ, does not allow $\mu = 0$.

Comparing FG and SG Methods

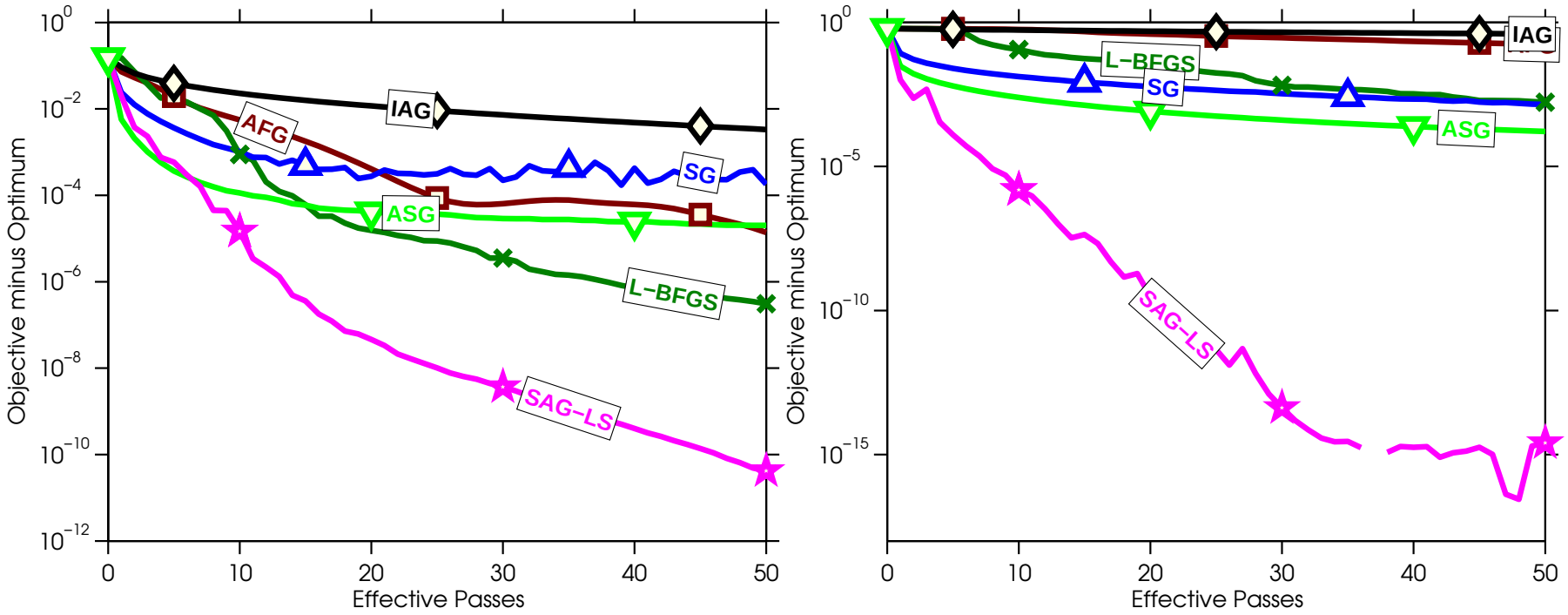
- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



- Comparison of competitive deterministic and stochastic methods.

SAG Compared to FG and SG Methods

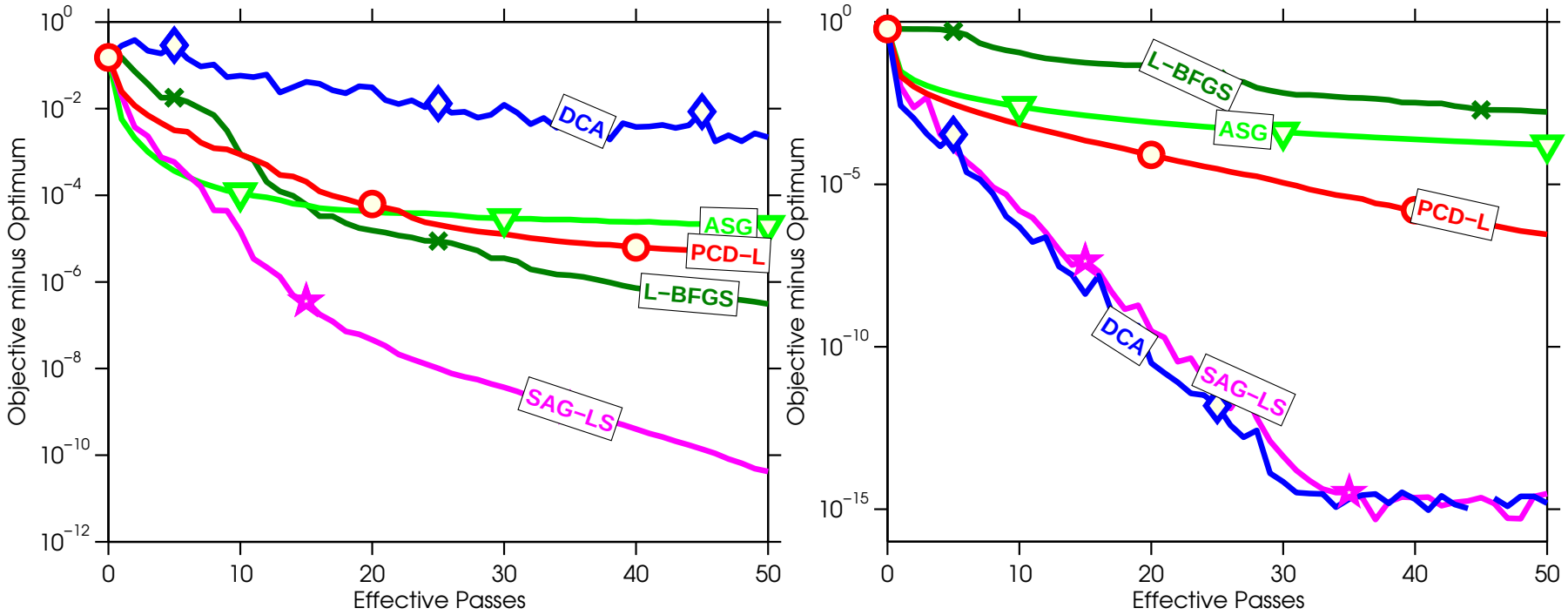
- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



- SAG starts fast and stays fast.

SAG Compared to Coordinate-Based Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



- PCD/DCA are similar on some problems, much worse on others.

SAG Implementation Issues

- while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.

SAG Implementation Issues: Initialization

- while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.
- The y_i may be initialized to zero.

SAG Implementation Issues: Initialization

- while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{M} d$.
- The y_i may be initialized to zero.
- We normalize by number of examples seen (M).
- Similar to doing one pass of SG.

SAG Implementation Issues: Termination

- while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{M} d$.
- When should we stop?
- Normally we check the size of $\|f'(x)\|$.

SAG Implementation Issues: Termination

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{M} d$.
- When should we stop?
- Normally we check the size of $\|f'(x)\|$.
- Since $y_i \to f_i'(x)$, check $\frac{1}{N}\|d\| = \|\frac{1}{N}\sum_{i=1}^{N} y_i\| \to \|f'(x)\|$

SAG Implementation Issues: Step Size

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$. .
  - $x = x - \frac{\alpha}{M} d$.
- How should we set the step size?

SAG Implementation Issues: Step Size

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$. .
  - $x = x - \frac{1}{LM}d$.
- How should we set the step size?
  - Theory: $\alpha = 1/16L$, Practice: $\alpha = 1/L$.

SAG Implementation Issues: Step Size

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$. .
  - $x = x - \frac{1}{LM}d$.
- How should we set the step size?
  - Theory: $\alpha = 1/16L$, Practice: $\alpha = 1/L$.
- If L is unknown

SAG Implementation Issues: Step Size

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $L = \text{lineSearch}(f_i, L)$.
  - $x = x - \frac{1}{LM}d$.
- How should we set the step size?
  - Theory: $\alpha = 1/16L$, Practice: $\alpha = 1/L$.
- If L is unknown or smaller near x^*, increase L until:

$$f_i(x^+) \leq f_i'(x) + \langle f_i'(x), x^+ - x\rangle + \frac{L}{2}\|x^+ - x\|, \text{ with } x^+ = x - \frac{1}{L}f_i'(x).$$

(Lipschitz approximation procedure from FISTA)

SAG Implementation Issues: Step Size

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $L = \text{lineSearch}(f_i, L)$.
  - $x = x - \frac{1}{LM}d$.
- How should we set the step size?
  - Theory: $\alpha = 1/16L$, Practice: $\alpha = 1/L$.
- If L is unknown or smaller near x^*, increase L until:

$$f_i(x^+) \leq f_i'(x) + \langle f_i'(x), x^+ - x\rangle + \frac{L}{2}\|x^+ - x\|, \text{ with } x^+ = x - \frac{1}{L}f_i'(x).$$

  (Lipschitz approximation procedure from FISTA)
- Decrease L between iterations, $L = L2^{-\frac{1}{N}}$.

SAG Implementation Issues: Reducing Memory

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $L = \text{lineSearch}(f_i, y_i, L)$.
  - $x = x - \frac{1}{LM}d$.
- Can we reduce the memory if $f_i'(x)$ is not sparse?
- For $f_i(a_i^T x)$ (e.g., least squares), use that $f_i'(a_i^T x) = a_i f_i'(\delta)$.

SAG Implementation Issues: Reducing Memory

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(\delta)$, with $\delta = a_i^T x$.
  - $d = d - a_i(y_i - f_i'(\delta)$.
  - $y_i = \delta$.
  - $L = \text{lineSearch}(f_i, y_i, L)$.
  - $x = x - \frac{1}{LM} d$.
- Can we reduce the memory if $f_i'(x)$ is not sparse?
- For $f_i(a_i^T x)$ (e.g., least squares), use that $f_i'(a_i^T x) = a_i f_i'(\delta)$.
- Only store the δ values to reduce memory from $O(NP)$ to $O(N)$.

SAG Implementation Issues: Reducing Memory

- while($\frac{1}{N}\|d\| \leq \epsilon$)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(\delta)$, with $\delta = a_i^T x$.
  - $d = d - a_i(y_i - f_i'(\delta)$.
  - $y_i = \delta$.
  - $L = \text{lineSearch}(f_i, y_i, L)$.
  - $x = x - \frac{1}{LM} d$.
- Can we reduce the memory if $f_i'(x)$ is not sparse?
- For $f_i(a_i^T x)$ (e.g., least squares), use that $f_i'(a_i^T x) = a_i f_i'(\delta)$.
- Only store the δ values to reduce memory from $O(NP)$ to $O(N)$.
- Line-search is $O(1)$ in N and P.
- Standard tricks avoid full-vector operations, allow regularizers.

SAG Implementation Issues: Mini-Batches

- Can we use mini-batches?

SAG Implementation Issues: Mini-Batches

- Can we use mini-batches?
  - Yes, define each f_i to include more than one example.
  - Reduces memory requirements.
  - Allows parallelization.
  - But must decrease L for good performance

$$L_{\mathcal{B}} \leq \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} L_i \leq \max_{i \in \mathcal{B}} \{L_i\}.$$

SAG Implementation Issues: Mini-Batches

- Can we use mini-batches?
  - Yes, define each f_i to include more than one example.
  - Reduces memory requirements.
  - Allows parallelization.
  - But must decrease L for good performance

$$L_{\mathcal{B}} \leq \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} L_i \leq \max_{i \in \mathcal{B}} \{L_i\}.$$

  - In practice, use the line-search on the batch to determine $L_{\mathcal{B}}$.

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!
  - Performance is intermediate between IAG and SAG.

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - Bias sampling towards Lipschitz constants L_i.

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - Bias sampling towards Lipschitz constants L_i.
  - Justification: duplicate examples proportional to L_i:

$$\frac{1}{N}\sum_{i=1}^{N} f_i(x) = \frac{1}{\sum L_i}\sum_{i=1}^{N}\sum_{j=1}^{L_i} L_{\text{mean}}\frac{f_i(x)}{L_i},$$

  convergence rate depends on L_{mean} instead of $L_{\max}$.

SAG Implementation Issues: Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - NO!
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - Bias sampling towards Lipschitz constants L_i.
  - Justification: duplicate examples proportional to L_i:

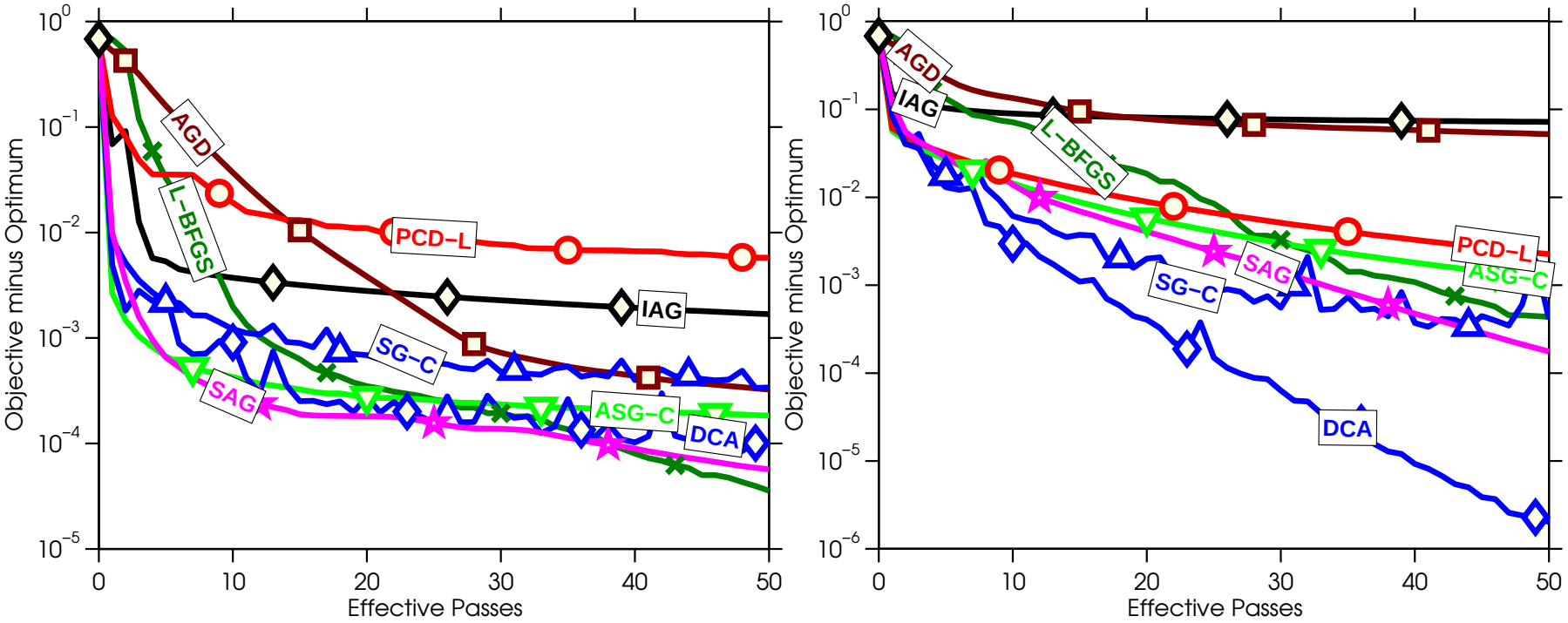
$$\frac{1}{N}\sum_{i=1}^{N} f_i(x) = \frac{1}{\sum L_i}\sum_{i=1}^{N}\sum_{j=1}^{L_i} L_{\text{mean}}\frac{f_i(x)}{L_i},$$

    convergence rate depends on L_{mean} instead of $L_{\max}$.
  - Combine with the line-search for adaptive sampling.

    (see paper/code for details)

SAG with Non-Uniform Sampling

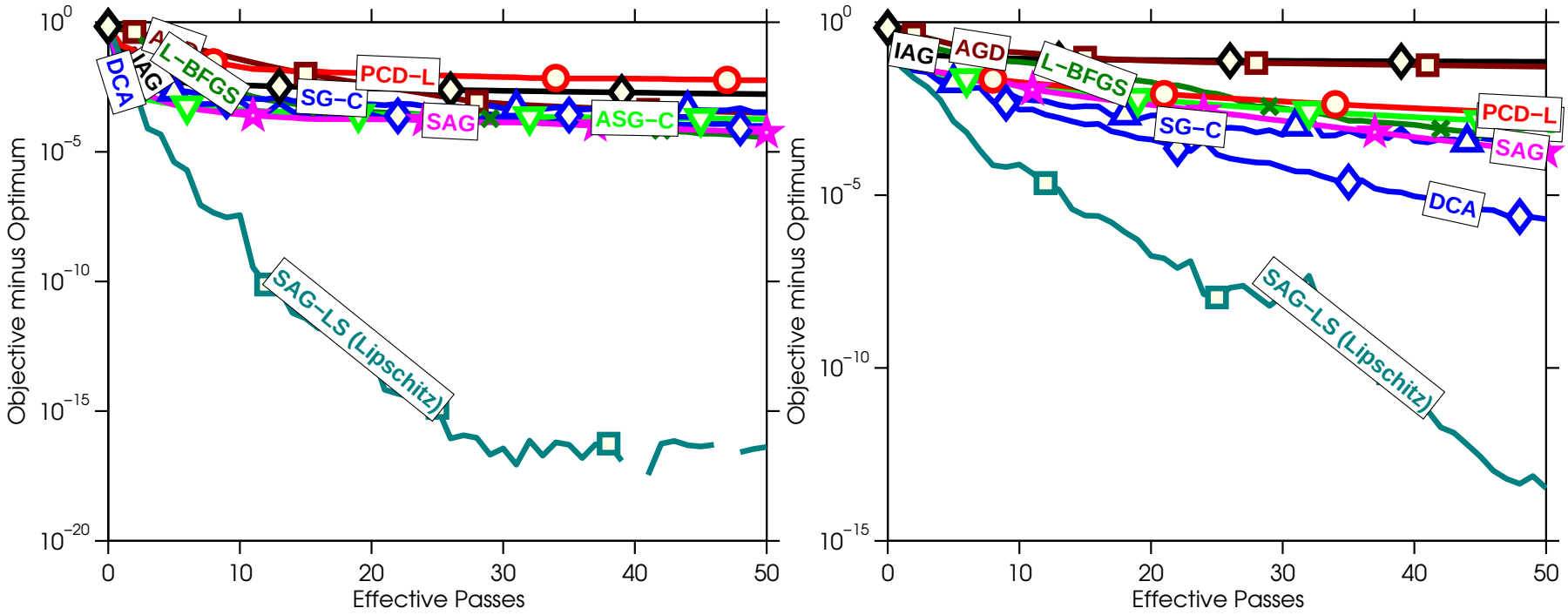
- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)



- Datasets where SAG had the worst relative performance.

SAG with Non-Uniform Sampling

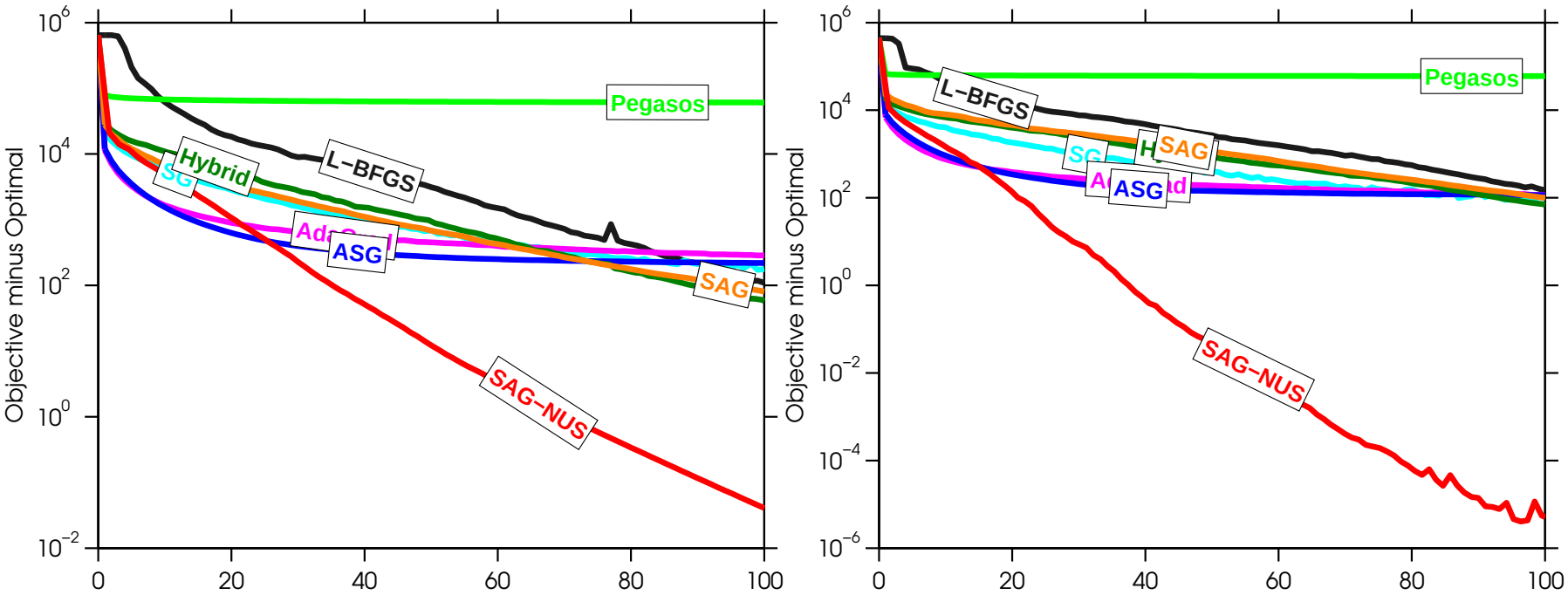
- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)



- Lipschitz sampling helps a lot.

SAG with Non-Uniform Sampling

- Noun-phrase chunking and named-entity recognition.



Conclusion and Discussion

- Faster theoretical convergence using only the ‘sum’ structure.

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.
- Constrained and non-smooth problems:
  - Proximal-gradient, ADMM.

  [Mairal, 2013, 2014, Wong et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014]

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.
- Constrained and non-smooth problems:
  - Proximal-gradient, ADMM.

  [Mairal, 2013, 2014, Wong et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014]
- Memory-free methods:
  - Similar performance, but requires two f_i' evaluations per iteration.

  [Mahdavi et al., 2013, Johnson and Zhang, 2013, Zhang et al., 2013, Konecny and Richtarik, 2013, Xiao and Zhang, 2014]

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.
- Constrained and non-smooth problems:
  - Proximal-gradient, ADMM.

  [Mairal, 2013, 2014, Wong et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014]
- Memory-free methods:
  - Similar performance, but requires two f_i' evaluations per iteration.

  [Mahdavi et al., 2013, Johnson and Zhang, 2013, Zhang et al., 2013, Konecny and Richtarik, 2013, Xiao and Zhang, 2014]
- Quasi-Newton methods:
  - Empirically faster convergence, but much more overhead.

  [Sohl-Dickstein et al., 2014]

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.
- Constrained and non-smooth problems:
  - Proximal-gradient, ADMM.

  [Mairal, 2013, 2014, Wong et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014]
- Memory-free methods:
  - Similar performance, but requires two f_i' evaluations per iteration.

  [Mahdavi et al., 2013, Johnson and Zhang, 2013, Zhang et al., 2013, Konecny and Richtarik, 2013, Xiao and Zhang, 2014]
- Quasi-Newton methods:
  - Empirically faster convergence, but much more overhead.

  [Sohl-Dickstein et al., 2014]
- Parallel/distributed methods.

Conclusion and Discussion

- Faster theoretical convergence using only the 'sum' structure.
- Simple algorithm, empirically better than theory predicts.
- Black-box stochastic gradient algorithm:
  - Adaptivity to problem difficulty, line-search, termination criterion.
- Constrained and non-smooth problems:
  - Proximal-gradient, ADMM.

  [Mairal, 2013, 2014, Wong et al., 2013, Xiao and Zhang, 2014, Defazio et al., 2014]
- Memory-free methods:
  - Similar performance, but requires two f_i' evaluations per iteration.

  [Mahdavi et al., 2013, Johnson and Zhang, 2013, Zhang et al., 2013, Konecny and Richtarik, 2013, Xiao and Zhang, 2014]
- Quasi-Newton methods:
  - Empirically faster convergence, but much more overhead.

  [Sohl-Dickstein et al., 2014]
- Parallel/distributed methods.
- Thank you for the invitation.