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 onSparse Representations, Numerical Linear Algebra, and Optimization
We want to minimize the sum of a finite set of smooth functions:

\[
\min_{x \in \mathbb{R}^P} g(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x).
\]
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 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 \( \ell_2 \)-regularized least-squares,

\[
f_i(x) := (a_i^T x - b_i)^2 + \frac{\lambda}{2} \|x\|^2.
\]
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 \( \ell_2 \)-regularized least-squares,

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

Other examples include any \( \ell_2 \)-regularized convex loss:
- logistic regression, Huber regression, smooth SVMs, CRFs, etc.
We consider minimizing \( g(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x) \).
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 \( O(N) \).
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: $O(1/t)$. 

Mark Schmidt

Minimizing Finite Sums with the SAG Algorithm
We consider minimizing $$g(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$.  

- **Deterministic** gradient method [Cauchy, 1847]:
  - Batch gradient descent:
    $$\theta^t = \theta^t - \frac{1}{n} \sum_{i=1}^{n} g'(\theta^t - 1)$$
  - Stochastic gradient descent:
    $$\theta^t = \theta^t - \gamma^t f'_i(t)(\theta^t - 1)$$

- **Stochastic** gradient method [Robbins & Monro, 1951]:
  - Stochastic gradient descent: $$\theta^t = \theta^t - \frac{1}{n} \sum_{i=1}^{n} g'(\theta^t - 1)$$
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.

Stochastic vs. deterministic methods
- **Goal** = best of both worlds: linear rate with $O(1)$ iteration cost

hybrid

log(excess cost)

stochastic
deterministic

hybrid

time

Mark Schmidt
Minimizing Finite Sums with the SAG Algorithm
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.

**Stochastic vs. deterministic methods**

- **Goal** = best of both worlds: linear rate with $O(1)$ iteration cost

- **Hybrid** cost: $\log(\text{excess cost})$

   - **Stochastic**
   - **Deterministic**
   - **Hybrid**

- **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**

- **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)]
A variety of methods have been proposed to speed up SG methods:

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

- **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**
Existing linear convergence results:

- **Constant step-size SG, accelerated SG**
  
  \[ [\text{Kesten (1958), Delyon and Juditsky (1993), Nedic and Bertsekas (2000)}] \]
  
  - 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
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
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$-continuous, $g$ is $\mu$-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 $\{1, 2, \ldots, n\}$ and compute $f'_i(x_t)$. 

$x_{t+1} = x_t - \alpha_t \sum_{i=1}^n f'_i(x_t)$

Memory: $y_{ti} = f'_i(x_t)$ from the last iteration $t$ where $i$ was selected.

Assumes that gradients of other examples don't change. This assumption becomes accurate as $\|x_{t+1} - x_t\| \rightarrow 0$. 

Stochastic variant of increment aggregated gradient (IAG). 

[Blatt et al. 2007]
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$-continuous, $g$ is $\mu$-strongly convex.

Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, $g$ is $\mu$–strongly convex.

Is it possible to have an $O(\rho^t)$ rate with an $O(1)$ cost?

YES!
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, $g$ is $\mu$-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 $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} f'_i(x^t)$$
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, $g$ is $\mu$-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 $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

\[
x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y^t_i
\]

Memory: $y^t_i = f'_{i(t)}(x^t)$ from the last iteration $t$ where $i$ was selected.

Assumes that gradients of other examples don’t change. This assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, $g$ is $\mu$-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 $\{1, 2, \ldots, n\}$ 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.
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, $g$ is $\mu$-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 $\{1, 2, \ldots, n\}$ and compute $f'_{i(t)}(x^t)$.

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

- **Memory:** $y^t_i = f'_{i(t)}(x^t)$ 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]
Proposition 1. With $\alpha_t = \frac{1}{2nL}$ the SAG iterations satisfy

$$\mathbb{E}[g(x^t) - g(x^*)] \leq \left(1 - \frac{\mu}{8LN}\right)^t C.$$
Proposition 1. With $\alpha_t = \frac{1}{2nL}$, the SAG iterations satisfy

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

- Convergence rate of $O(\rho^t)$ with cost of $O(1)$.
- Mission accomplished?!?
Proposition 1. With $\alpha_t = \frac{1}{2nL}$ the SAG iterations satisfy

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

- Convergence rate of $O(\rho^t)$ with cost of $O(1)$.
- Mission accomplished?!!
- This rate is very slow: performance similar to cyclic method.
Proposition 2. With $\alpha_t \in \left[ \frac{1}{2n\mu}, \frac{1}{16L} \right]$ and $N \geq 8 \frac{L}{\mu}$, the SAG iterations satisfy

$$
\mathbb{E}[g(x^t) - g(x^*)] \leq \left( 1 - \frac{1}{8N} \right)^t C.
$$
Proposition 2. With $\alpha_t \in \left[ \frac{1}{2n\mu}, \frac{1}{16L} \right]$ and $N \geq \frac{8L}{\mu}$, the SAG iterations satisfy

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

- Much bigger step-sizes: $\mu << L$ (this does not work for cyclic)
Proposition 2. With $\alpha_t \in \left[\frac{1}{2n\mu}, \frac{1}{16L}\right]$ and $N \geq 8\frac{L}{\mu}$, the SAG iterations satisfy

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

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

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

- $N \geq O\left(\frac{L}{\mu}\right)$ has been called ‘big data’ condition.
Theorem. With $\alpha_t = \frac{1}{16L}$ the SAG iterations satisfy

$$\mathbb{E}[g(x^t) - g(x^*)] \leq \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C.$$
**Theorem.** With $\alpha_t = \frac{1}{16L}$ the SAG iterations satisfy

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

- This rate is “very fast”:
  - Well-conditioned problems: constant non-trivial reduction per pass.
Theorem. With $\alpha_t = \frac{1}{16L}$ the SAG iterations satisfy

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

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

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

with $\alpha_t = \frac{1}{L}$, but SAG is $N$ times faster.
Assume that \( N = 700000, L = 0.25, \mu = 1/N \):

- Gradient method has rate \((L - \mu L + \mu)^2 = 0.99998\).
- Accelerated gradient method has rate \((1 - \sqrt{\mu L}) = 0.99761\).
- SAG (\(N\) iterations) has rate \((1 - \min\{\mu 16L, 1/8N\}) = 0.88250\).

Fastest possible first-order method:

\[ (\sqrt{L} - \sqrt{\mu} \sqrt{L} + \sqrt{\mu})^2 = 0.99048. \]

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

Number of \(f_i'\) evaluations to reach \(\epsilon\):

- Stochastic: \(O(L \mu (1/\epsilon))\).
- Gradient: \(O(N L \mu \log(1/\epsilon))\).
- Accelerated: \(O(N \sqrt{L \mu \log(1/\epsilon)})\).
- SAG: \(O(\max\{N, L \mu\} \log(1/\epsilon))\).
Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:

Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$. 

SAG ($N$ iterations) has rate $\left(1 - \min\{\mu \frac{16}{L}, \frac{1}{8N}\}\right)N = 0.88250$. 

Fastest possible first-order method: $\left(\sqrt{L} - \sqrt{\mu} \sqrt{L} + \sqrt{\mu}\right)^2 = 0.99048$. 

SAG beats two lower bounds: Stochastic gradient bound (of $O\left(\frac{1}{t}\right)$). Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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: $\left(\sqrt{L} - \sqrt{\mu} \sqrt{L} + \sqrt{\mu}\right)^2 = 0.99048$.

SAG beats two lower bounds: 
- Stochastic gradient bound (of $\mathcal{O}\left(\frac{1}{t}\right)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- SAG ($N$ iterations) has rate $(1 - \min \left\{ \frac{\mu}{16L}, \frac{1}{8N} \right\})^N = 0.88250$. 
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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- SAG ($N$ iterations) has rate $(1 - \min \left\{ \frac{\mu}{16L}, \frac{1}{8N} \right\})^N = 0.88250$.
- 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 $O\left(\frac{1}{t}\right)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).
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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- 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 $L$, $\mu$, 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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- SAG ($N$ iterations) has rate $(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\})^N = 0.88250$.
- *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 $L$, $\mu$, and $N$).

Number of $f'_i$ evaluations to reach $\epsilon$: 

Mark Schmidt
Minimizing Finite Sums with the SAG Algorithm
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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
  - 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 $L$, $\mu$, and $N$).

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

- Number of $f'_i$ evaluations to reach $\epsilon$:
  - 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 $(1 - \sqrt{\frac{\mu}{L}}) = 0.99761$.
- SAG ($N$ iterations) has rate $(1 - \min \left\{ \frac{\mu}{16L}, \frac{1}{8N} \right\})^N = 0.88250$.
- Fastest possible first-order method: $\left(\frac{\sqrt{L-\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

SAG beats two lower bounds:

- Stochastic gradient bound (of $O(1/t)$).
- Deterministic gradient bound (for typical $L$, $\mu$, and $N$).

Number of $f_i'$ evaluations to reach $\epsilon$:

- Stochastic: $O\left(\frac{L}{\mu} \frac{1}{\epsilon}\right)$.
- Gradient: $O(N\frac{L}{\mu} \log(1/\epsilon))$.
- Accelerated: $O(N\sqrt{\frac{L}{\mu}} \log(1/\epsilon))$. 
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 \( 1 - \sqrt{\frac{\mu}{L}} = 0.99761 \).
- SAG ($N$ iterations) has rate \( (1 - \min \{ \frac{\mu}{16L}, \frac{1}{8N} \})^N = 0.88250 \).
- 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 $L$, $\mu$, and $N$).

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

\[ \begin{array}{c}
\theta^t \\
\theta^* \\
e
\end{array} = 
\begin{bmatrix}
y_1^t \\
\vdots \\
y_N^t \\
x^t
\end{bmatrix}, 
\begin{array}{c}
f'_i(x^*) \\
\vdots \\
f'_N(x^*) \\
x^*
\end{array}, 
\begin{bmatrix}
I \\
\vdots \\
I
\end{bmatrix},
\]

\[ A = a_1 ee^\top + a_2 I, \quad B = be, \quad C = cl. \]
Proof Technique: Lyapunov Function

- We define a Lyapunov function of the form

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

with

\[
\begin{align*}
\theta^t &= \begin{bmatrix} y_1^t \\ \vdots \\ y_N^t \\ x^t \end{bmatrix}, \\
\theta^* &= \begin{bmatrix} f'_i(x^*) \\ \vdots \\ f'_N(x^*) \end{bmatrix}, \\
e &= \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, \\
A &= a_1 ee^T + a_2 I, \\
B &= be, \\
C &= cI.
\end{align*}
\]

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

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

- Apply recursively and initial Lyapunov function gives constant.
What are the constants?
What are the constants?

If we initialize with $y_i^0 = 0$ we have

$$C = [g(x^0) - g(x^*)] + \frac{4L}{N} \|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$
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
  \[
  C = \frac{3}{2} [g(x^0) - g(x^*)] + \frac{4L}{N} \|x^0 - x^*\|^2.
  \]
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,
  \[ [g(x^0) - g(x^*)] = O(1/N), \quad \|x^0 - x^*\|^2 = O(1/N). \]
Assume only that:

- $f_i$ is convex, $f'_i$ is $L$–continuous, some $x^*$ exists.
Assume only that:

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

\[ \text{Theorem. } \text{With } \alpha_t \leq \frac{1}{16L} \text{ the SAG iterations satisfy} \]

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

- Faster than SG lower bound of $O(1/\sqrt{t})$. 
Assume only that:

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

**Theorem.** With $\alpha_t \leq \frac{1}{16L}$ 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 $\mu$ is locally bigger around $x^*$.
Assume only that:

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

**Theorem.** With $\alpha_t \leq \frac{1}{16L}$ 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 $\mu$ 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 $\alpha_t \leq \frac{1}{16L}$ 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 $\mu$ 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 $\mu$, does not allow $\mu = 0$. 

Mark Schmidt
Minimizing Finite Sums with the SAG Algorithm
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.

\[\text{Objective minus Optimum vs. Effective Passes}\]
quantum \((n = 50000, p = 78)\) and rcv1 \((n = 697641, p = 47236)\)

PCD/DCA are similar on some problems, much worse on others.
while(1)

- Sample $i$ from $\{1, 2, \ldots, N\}$.
- Compute $f'_i(x)$.
- $d = d - y_i + f'_i(x)$.
- $y_i = f'_i(x)$.
- $x = x - \frac{\alpha}{N} d$. 
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.
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.
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 $\|f'(x)\|$. 
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: $\alpha = 1/L$. 
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
while\(\left(\frac{1}{N}\|d\| \leq \epsilon\right)\)

- 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)
while \( \left( \frac{1}{N} \| d \| \leq \epsilon \right) \)

- 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, \( L = L2^{-\frac{1}{N}} \).
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 $f'_i(a_i^T x) = a_i f'_i(\delta)$. 

Mark Schmidt
Minimizing Finite Sums with the SAG Algorithm
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 $\delta$ values to reduce memory from $O(NP)$ to $O(N)$. 
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 $\delta$ 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.
Can we use mini-batches?

Yes, define each $f_i$ to include more than one example. Reduces memory requirements. Allows parallelization. But must decrease $L_B \leq \frac{|B|}{\sum_{i \in B} L_i} \leq \max_{i \in B} \{ L_i \}$.

In practice, use the line-search on the batch to determine $L_B$. 
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

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

\[
L_B \leq \frac{1}{|B|} \sum_{i \in B} L_i \leq \max_{i \in B} \{L_i\}.
\]

- In practice, use the line-search on the batch to determine \( L_B \).
Does re-shuffling and doing full passes work better?

**Performance is intermediate between IAG and SAG.**

**Can non-uniform sampling help?**

Bias sampling towards Lipschitz constants $L_i$. Justification: duplicate examples proportional to $\frac{1}{N} \sum_i f_i(x) = \frac{1}{\sum_i L_i N} \sum_i = \frac{1}{L_i \sum_j L_i \text{mean} f_i(x)}$, convergence rate depends on $L_{\text{mean}}$ instead of $L_{\text{max}}$.

Combine with the line-search for adaptive sampling. (see paper/code for details)
Does re-shuffling and doing full passes work better?

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

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

Can non-uniform sampling help?
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 $\frac{1}{N} \sum_i f_i(x) = \frac{1}{\sum_i L_i} \sum_i \frac{1}{N} L_i \sum_j = \frac{1}{L_{\text{mean}}} f_i(x)$, convergence rate depends on $L_{\text{mean}}$ instead of $L_{\text{max}}$. Combine with the line-search for adaptive sampling. (see paper/code for details)
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 $L_{\text{mean}}$ instead of $L_{\text{max}}$. 

Mark Schmidt
Minimzing Finite Sums with the SAG Algorithm
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 $L_i$:

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

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

- Combine with the line-search for adaptive sampling.

(see paper/code for details)
Datasets where SAG had the worst relative performance.

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.

Mark Schmidt
Minimizing Finite Sums with the SAG Algorithm
- 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.

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.


**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.
  
- **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.
- 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.
- 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.