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

\[
\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).
\]
We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

We are interested in cases where $n$ is very large.

We will focus on strongly-convex functions $g$:

- Any convex function plus L2-regularization.
We want to minimize the sum of a finite set of smooth functions:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

We are interested in cases where $n$ is very large.

We will focus on strongly-convex functions $g$:
- Any convex function plus L2-regularization.

Simplest example is $\ell_2$-regularized least-squares,

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

Common framework in machine learning:
- logistic regression, Huber regression, smooth SVMs, CRFs, etc.
We consider minimizing $f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$.

**Deterministic gradient method** [Cauchy, 1847]:

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

Linear convergence rate: $O(\rho^t)$.

Iteration cost is linear in $n$.

Fancier methods exist, but still cost $O(n)$.

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

Random selection of $i_t$ from \{1, 2, ..., $N$\},

$$x_{t+1} = x_t - \alpha_t f'_{i_t}(x_t)$$

Iteration cost is independent of $n$.

Sublinear convergence rate: $O\left(\frac{1}{t}\right)$.
Stochastic vs. Deterministic Gradient Methods

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

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

  - Linear convergence rate: $O(\rho^t)$.
  - Iteration cost is linear in $n$.
  - Fancier methods exist, but still cost $O(n)$.
Stochastic vs. Deterministic Gradient Methods

- We consider minimizing \( f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \).
- **Deterministic** gradient method [Cauchy, 1847]:
  \[
  x_{t+1} = x_t - \alpha_t f'(x_t) = x_t - \frac{\alpha_t}{n} \sum_{i=1}^{n} f'_i(x_t).
  \]
  - **Linear** convergence rate: \( O(\rho^t) \).
  - Iteration cost is linear in \( n \).
  - Fancier methods exist, but still cost \( O(n) \)

- **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]:

\[
\theta_t = \theta_t - \gamma_t \nabla g(\theta_t - 1) = \theta_t - \gamma_t \frac{1}{n} \sum_{i=1}^{n} f_i'(\theta_t - 1)
\]

- **Stochastic gradient descent**:

\[
\theta_t = \theta_t - \gamma_t f_i'(t)(\theta_t - 1)
\]

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

\[
\theta_t = \theta_t - \gamma_t \nabla g(\theta_t - 1) = \theta_t - \gamma_t \frac{1}{n} \sum_{i=1}^{n} f_i'(\theta_t - 1)
\]
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
• $\log$ (excess cost)

It is possible to have linear rate with $O(1)$ cost?
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

It is possible to have linear rate with $O(1)$ cost?
Stochastic average gradient (SAG): [Le Roux et al., 2012]:

Randomly select $i_t$ from \{1, 2, \ldots, n\} and compute $f'_i(x_t)$,

$$x_{t+1} = x_t - \frac{\alpha_t}{n} \sum_{i=1}^{n} y^t_i,$$

where $y^t_i = f'_i$ from last iteration $s$ where $i$ was selected.
Stochastic Average Gradient (SAG)

- Stochastic average gradient (SAG): [Le Roux et al., 2012]:
  - Randomly select $i_t$ from $\{1, 2, \ldots, n\}$ and compute $f'_{i_t}(x_t)$,

\[
x_{t+1} = x_t - \frac{\alpha_t}{n} \sum_{i=1}^{n} y^t_i,
\]

where $y^t_i = f'_{i_s}$ from last iteration $s$ where $i$ was selected.

- Achieves $O(\rho^t)$ convergence rate with $O(1)$ iteration cost:
- Number of $f'_i$ evaluations to reach accuracy of $\epsilon$:
  - Stochastic gradient: $O(\kappa/\epsilon)$.
  - Deterministic gradient: $O(n\kappa \log(1/\epsilon))$.
  - Accelerated gradient: $O(n\sqrt{\kappa} \log(1/\epsilon))$.
  - Stochastic average gradient: $O((n + \kappa) \log(1/\epsilon))$. 

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.
Other methods subsequently shown to have this property:

- SDCA [Shalev-Schwartz & Zhang, 2013].
- MISO [Mairal, 2013].
- SAGA [Defazio et al., 2014].
Other methods subsequently shown to have this property:

- SDCA [Shalev-Schwartz & Zhang, 2013].
- MISO [Mairal, 2013].
- SAGA [Defazio et al., 2014].

But, these all introduce memory requirements:

- Require previous gradients $f_i'$ or dual variables for each $i$.
- Only $O(n)$ for some objectives, but $O(nd)$ in general.
Recent methods with similar rates that **avoid memory:**

- Mixed Gradient [Mahdavi & Jin, 2013, Zhang et al., 2013]
- Stochastic variance-reduced gradient (**SVRG**) [Johnson & Zhang, 2013]
- Semi-stochastic gradient [Konecny & Richtarik, 2013]
Recent methods with similar rates that avoid memory:
  - Mixed Gradient [Mahdavi & Jin, 2013, Zhang et al., 2013]
  - Stochastic variance-reduced gradient (SVRG) [Johnson & Zhang, 2013]
  - Semi-stochastic gradient [Konecny & Richtarik, 2013]

Memory is $O(d)$, but they require extra gradient calculations:
  - Two gradients on each iteration.
  - Occasional calculation of all $n$ gradients.

Extra calculations make them slower than SAG and friends.
1. Deterministic, stochastic, and finite-sum methods.
2. Wasting fewer gradients in SVRG.
SVRG algorithm (SG method with control variate):

- Start with $x_0$
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f'_i(x_s)$ (full gradient calculation)
- $x_{s+1} = x_t - \alpha_t (f'_i(x_t - 1) - f'_i(x_s) + d_s)$
  (inner loop)
- $x_0 = x_s$ for random $t \in \{1, 2, \ldots, m\}$ (initialize next outer loop)
- Only need to store $x_s$ and $d_s$. (outer loop)
SVRG algorithm (SG method with control variate):

- Start with $x_0$
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f'_i(x_s)$  
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick $i_t \in \{1, 2, \ldots, n\}$
    - $x^t = x^{t-1} - \alpha_t(f'_{i_t}(x^{t-1}) - f'_{i_t}(x_s) + d_s)$
SVRG algorithm (SG method with control variate):

- Start with $x_0$
- for $s = 0, 1, 2 \ldots$
  
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f'_i(x_s)$
  
  - $x^0 = x_s$
  
    - for $t = 1, 2, \ldots m$
      
        - Randomly pick $i_t \in \{1, 2, \ldots, n\}$
        
        - $x^t = x^{t-1} - \alpha_t(f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$

- $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$

Only need to store $x_s$ and $d_s$. 
Convergence Analysis of SVRG

- Assumptions:
  - Each $f_i$ is convex.
  - Each $f'_i$ is $L$-Lipschitz continuous.
  - Average $f$ is $\mu$-strongly convex.

Johnson & Zhang [2013] show that outer loop satisfies

$$E[f(x_s+1) - f(x^*)] \leq \rho(L,L)[f(x_s) - f(x^*)],$$

where $ho(a,b) = \frac{1}{1 - 2\alpha a (2b\alpha + 1)m\mu}$. SVRG rate is very fast for appropriate $\alpha$ and $m$.

In practice:

- $m = n$ (alternate between computing gradient and stochastic pass).
- $\alpha = \frac{1}{L}$ (slightly larger than allowed by theory).
- $x_{s+1} = x_m$ (rather than random).
Assumptions:

- Each $f_i$ is convex.
- Each $f'_i$ is $L$-Lipschitz continuous.
- Average $f$ is $\mu$-strongly convex.

Johnson & Zhang [2013] show that outer loop satisfies

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L, L)[f(x^s) - f(x^*)],$$

where

$$\rho(a, b) = \frac{1}{1 - 2\alpha a \left(2b\alpha + \frac{1}{m\mu\alpha}\right)}.$$ 

SVRG rate is very fast for appropriate $\alpha$ and $m$. 

In practice:

- $m = n$ (alternate between computing gradient and stochastic pass).
- $\alpha = 1/L$ (slightly larger than allowed by theory).
- $x^{s+1} = x^m$ (rather than random).
Convergence Analysis of SVRG

- **Assumptions:**
  - Each $f_i$ is convex.
  - Each $f'_i$ is $L$-Lipschitz continuous.
  - Average $f$ is $\mu$-strongly convex.

- Johnson & Zhang [2013] show that outer loop satisfies

  $$
  \mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L, L)[f(x^s) - f(x^*)],
  $$

  where

  $$
  \rho(a, b) = \frac{1}{1 - 2\alpha a} \left( 2b\alpha + \frac{1}{m\mu\alpha} \right).
  $$

- SVRG rate is very fast for appropriate $\alpha$ and $m$.

- In practice:
  - $m = n$ (alternate between computing gradient and stochastic pass).
  - $\alpha = 1/L$ (slightly larger than allowed by theory).
  - $x^{s+1} = x_m$ (rather than random).
We first give a result for SVRG with error:

Assume:

- We approximate full gradient by $d^s = f'(x^s) + e^s$.
- $\|x_t - x^*\| \leq Z$ for some $Z$. 
We first give a result for **SVRG with error**:

Assume:
- We approximate full gradient by $d^s = f'(x^s) + e^s$.
- $\|x_t - x^*\| \leq Z$ for some $Z$.

Then SVRG with error satisfies

$$
\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L, L)[f(x^s) - f(x^*)] + \frac{\alpha \mathbb{E}[\|e^s\|^2] + Z \mathbb{E}[\|e^s\|]}{1 - 2\alpha L}.
$$

Implications:
- Same convergence rate if $\max\{ \mathbb{E}[\|e^s\|], \mathbb{E}[\|e^s\|^2] \} = O(\tilde{\rho}s)$ for $\tilde{\rho} < \rho$.
- Tolerates large error when far from solution.
We first give a result for SVRG with error:

Assume:

- We approximate full gradient by \( d^s = f'(x^s) + e^s \).
- \( \|x_t - x^*\| \leq Z \) for some \( Z \).

Then SVRG with error satisfies

\[
\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L, L)[f(x^s) - f(x^*)] + \frac{\alpha\mathbb{E}[\|e^s\|^2] + Z\mathbb{E}[\|e^s\|]}{1 - 2\alpha L}.
\]

Implications:

- **Same convergence rate** if \( \max\{\mathbb{E}\|e^s\|, \mathbb{E}\|e^s\|^2\} = O(\bar{\rho}^s) \) for \( \bar{\rho} < \rho \).
- Tolerates large error when far from solution.
SVRG requires $2m + n$ gradients for each $m$ iterations.
- SVRG requires $2m + n$ gradients for each $m$ iterations.
- We can reduce the $n$ by using a ‘batch’ $B^s$ of training examples:

$$d^s = \frac{1}{|B^s|} \sum_{i \in B^s} f'_i(x^s).$$

- Special case of SVRG with error, batch size $|B^s|$ controls error.
Reducing Gradient Evaluations with Batching

- SVRG requires \(2m + n\) gradients for each \(m\) iterations.
- We can reduce the \(n\) by using a ‘batch’ \(B^s\) of training examples:
  \[
d^s = \frac{1}{|B^s|} \sum_{i \in B^s} f'_i(x^s).
\]
- Special case of SVRG with error, batch size \(|B^s|\) controls error.
- By sampling without replacement, we maintain rate if
  \[
  |B^s| \geq \frac{nS^2}{S^2 + nO(\rho^{2s})}.
  \]
Reducing Gradient Evaluations with Batching

- SVRG requires $2m + n$ gradients for each $m$ iterations.
- We can reduce the $n$ by using a ‘batch’ $B^s$ of training examples:

$$d^s = \frac{1}{|B^s|} \sum_{i \in B^s} f'_i(x^s).$$

- Special case of SVRG with error, batch size $|B^s|$ controls error.
- By sampling without replacement, we maintain rate if

$$|B^s| \geq \frac{nS^2}{S^2 + nO(\tilde{\rho}^{2s})}.$$  

- Hard to do in practice, but we know shape of optimal batch schedule...
Batch Schedule Needed for Linear Rate

[Aravkin et al, 2012]
Growing-batch reduces $n$ in the $2m + n$ cost of SVRG.
Growing-batch reduces $n$ in the $2m + n$ cost of SVRG.

But, does not improve the $2$:

- Important in early iterations when we reduce test error the most.

Convergence rate:

$$E[f(x_s + 1) - f(x^*)] \leq \rho \left( L, \frac{|B_s|}{n} L \right) E[f(x_s) - f(x^*)] + \alpha E[\|e_s\|_2^2] + \tilde{Z} E[\|e_s\|_1 - 2\alpha L + \alpha^2 \left( 1 - \frac{|B_s|}{n} \right) \sigma^2 \left( 1 - 2\alpha L \right)].$$

Improves rate when far from solution. But dependence on variance $\sigma^2$. 
Mixed SG and SVRG Method

- Growing-batch reduces $n$ in the $2m + n$ cost of SVRG.
- But, does not improve the $2^i$:
  - Important in early iterations when we reduce test error the most.
- To improve the $2^i$, consider a mixed strategy:
  - If $i$ is in the batch $B^s$, take SVRG step (2 gradients).
  - If $i$ is not in the batch, take SG step (1 gradient).
Growing-batch reduces $n$ in the $2m + n$ cost of SVRG.

But, does not improve the $2$:

- Important in early iterations when we reduce test error the most.

To improve the $2$, consider a mixed strategy:

- If $i$ is in the batch $B^s$, take SVRG step (2 gradients).
- If $i$ is not in the batch, take SG step (1 gradient).

Convergence rate:

$$
\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho \left( L, \frac{|B^s|}{n} L \right) [f(x^s) - f(x^*)] 
+ \frac{\alpha \mathbb{E} [\|e^s\|^2]}{1 - 2\alpha L} + Z \mathbb{E} [\|e^s\|] + \frac{\alpha (1 - |B^s|/n)\sigma^2}{2 (1 - 2\alpha L)}.
$$

- Improves rate when far from solution.
- But dependence on variance $\sigma^2$. 


Numerical Experiments with Batching

Training/testing loss for $\ell_2$-regularized logistic on spam filtering data.
Mixed strategy improves error when far from solution.

Consider Huberized hinge loss problem [Rosset & Zhu, 2006]:

\[
\min_{x \in \mathbb{R}^d} \sum_{i=1}^n f(b_i a_i^T x), \quad f(\tau) = \begin{cases} 0 & \text{if } \tau > 1 + \epsilon, \\ 1 - \tau & \text{if } \tau < 1 - \epsilon, \\ (1 + \epsilon - \tau)^2/4\epsilon & \text{if } |1 - \tau| \leq \epsilon. \end{cases}
\]

The solution is sparse in the support vectors.
Identifying Support Vectors

- Mixed strategy improves error when **far from solution**.
- For certain objectives, can improve **close to solution**.

Consider Huberized hinge loss problem [Rosset & Zhu, 2006]:

\[
\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} f(b_i a_i^T x),
\]

where

\[
f(\tau) = \begin{cases} 
0 & \text{if } \tau > 1 + \epsilon, \\
1 - \tau & \text{if } \tau < 1 - \epsilon, \\
\frac{(1 + \epsilon - \tau)^2}{4 \epsilon} & \text{if } |1 - \tau| \leq \epsilon.
\end{cases}
\]
Identifying Support Vectors

- Mixed strategy improves error when far from solution.
- For certain objectives, can improve close to solution.
- Consider Huberized hinge loss problem [Rosset & Zhu, 2006]:

\[
\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f(b_i a_i^T \mathbf{x}), \quad f(\tau) = \begin{cases} 
0 & \text{if } \tau > 1 + \epsilon, \\
1 - \tau & \text{if } \tau < 1 - \epsilon, \\
\frac{(1+\epsilon-\tau)^2}{4\epsilon} & \text{if } |1 - \tau| \leq \epsilon.
\end{cases}
\]

- The solution is sparse in the \( f_i' \) (has support vectors).
- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.
Using Support Vectors

- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.
- Approach 1 (sound pruning):
  - Maintain list of support vectors at $x_s$.

Related to shrinking heuristic in SVM solvers [Joachims, 1999, Usunier et al., 2010].
Non-support examples do not contribute to solution.

We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.

Approach 1 (sound pruning):
- Maintain list of support vectors at $x_s$.
- Do not evaluate $f'_i(x_s)$ if it is not a support vector.
- Can reduce number of gradients per iteration to 1.

Approach 2 (heuristic pruning):
- Keep track of the number of times we $f'_i(x_s) = 0$ or $f'_i(x_t) = 0$.
- If it's been zero more than once consecutively, skip its next evaluation.
- If it continues to be zero, skip its next 2 evaluations.
- If it continues to be zero, skip its next 4 evaluations.
- Can reduce number of gradients per iteration to 1 or 0.

Related to shrinking heuristic in SVM solvers [Joachims, 1999, Usunier et al., 2010].
Using Support Vectors

- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.
- Approach 1 (sound pruning):
  - Maintain list of support vectors at $x_s$.
  - Do not evaluate $f'_i(x_s)$ if it is not a support vector.
  - Can reduce number of gradients per iteration to 1.
- Approach 2 (heuristic pruning):
  - Keep track of the number of times we $f'_i(x_s) = 0$ or $f'_i(x^t) = 0$. 
  - If it's been zero more than once consecutively, skip its next evaluation.
  - If it continues to be zero, skip its next 2 evaluations.
  - If it continues to be zero, skip its next 4 evaluations.
  - Can reduce number of gradients per iteration to 1 or 0.

Related to shrinking heuristic in SVM solvers [Joachims, 1999, Usunier et al., 2010].
Using Support Vectors

- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.
- Approach 1 (sound pruning):
  - Maintain list of support vectors at $x_s$.
  - Do not evaluate $f'_i(x_s)$ if it is not a support vector.
  - Can reduce number of gradients per iteration to 1.

- Approach 2 (heuristic pruning):
  - Keep track of the number of times we $f'_i(x_s) = 0$ or $f'_i(x^t) = 0$.
  - If it's been zero more than once consecutively, skip its next evaluation.

Related to shrinking heuristic in SVM solvers [Joachims, 1999, Usunier et al., 2010].
Using Support Vectors

- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that $f'_i(x) = 0$.

**Approach 1 (sound pruning):**
- Maintain list of support vectors at $x_s$.
- Do not evaluate $f'_i(x_s)$ if it is not a support vector.
- Can reduce number of gradients per iteration to 1.

**Approach 2 (heuristic pruning):**
- Keep track of the number of times we $f'_i(x_s) = 0$ or $f'_i(x^t) = 0$.
- If it's been zero more than once consecutively, skip its next evaluation.
- If it continues to be zero, skip its next 2 evaluations.
Using Support Vectors

- Non-support examples do not contribute to solution.
- We can skip gradient evaluations where we expected/know that \( f_i'(x) = 0 \).
- Approach 1 (sound pruning):
  - Maintain list of support vectors at \( x_s \).
  - Do not evaluate \( f_i'(x_s) \) if it is not a support vector.
  - Can reduce number of gradients per iteration to 1.
- Approach 2 (heuristic pruning):
  - Keep track of the number of times we \( f_i'(x_s) = 0 \) or \( f_i'(x^t) = 0 \).
  - If it's been zero more than once consecutively, skip its next evaluation.
  - If it continues to be zero, skip its next 2 evaluations.
  - If it continues to be zero, skip its next 4 evaluations.
  - Can reduce number of gradients per iteration to 1 or 0.
- Related to shrinking heuristic in SVM solvers [Joachims, 1999, Usunier et al., 2010].
Numerical Experiments with Support Vectors

$l_2$-regularized Huberized hinge on spam filtering data.
1. Deterministic, stochastic, and finite-sum methods.
2. Wasting fewer gradients in SVRG.
Machine learning applications often have the form

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{\lambda}{2} \|x\|^2 + \frac{1}{n} \sum_{i=1}^N g_i(x).$$

The SVRG update has the form

$$x_t = x_{t-1} - \alpha_t \left(\lambda x_{t-1} - g'_i(x_{t-1}) - g'_i(x_t) + d_s\right),$$

which approximates

$$\sum_{i} g_i$$

and uses exact regularizer gradient:

$$x_t = \left(1 - \alpha_t \lambda\right) x_{t-1} - \alpha_t \left(g'_i(x_{t-1}) - g'_i(x_t) + d_s\right).$$

This form is nice for sparse implementation (also used in SAG/SAGA codes).

We show that the regularized update satisfies:

$$E[f(x_{s+1}) - f(x^*)] \leq \rho(L_m, L_m^2)[f(x_s) - f(x^*)],$$

where $L_m = \max\{\lambda, L_g\}$. SVRG actually converges faster than expected.
Sparse Gradients and L2-Regularization

- Machine learning applications often have the form

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{\lambda}{2} \|x\|^2 + \frac{1}{n} \sum_{i=1}^{N} g_i(x).$$

- The SVRG update has the form

$$x^t = x^{t-1} - \alpha_t((\lambda x^{t-1} + g_i'(x^{t-1})) - (\lambda x_s + g_i'(x_s)) + d_s),$$

which approximates $\sum_i g_i$ and uses exact regularizer gradient:

$$x^t = (1 - \alpha_t \lambda)x^{t-1} - \alpha_t(g_i'(x^{t-1}) - g_i'(x_s) + (d_g)_s),$$

- This form is nice for sparse implementation (also used in SAG/SAGA codes).
Machine learning applications often have the form

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{\lambda}{2} \|x\|^2 + \frac{1}{n} \sum_{i=1}^{N} g_i(x).$$

The SVRG update has the form

$$x^t = x^{t-1} - \alpha_t ((\lambda x^{t-1} + g'_i(x^{t-1})) - (\lambda x_s + g'_i(x_s)) + d_s),$$

which approximates $\sum_i g_i$ and uses exact regularizer gradient:

$$x^t = (1 - \alpha_t \lambda)x^{t-1} - \alpha_t (g'_i(x^{t-1}) - g'_i(x_s) + (d_g)_s),$$

This form is nice for sparse implementation (also used in SAG/SAGA codes).

We show that regularized update satisfies:

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L^m, L^m)[f(x^s) - f(x^*)],$$

where $L^m = \max\{\lambda, L_g\}$.

SVRG actually converges faster than expected.
A common non-smooth variation is solving problems of the form

$$\arg\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x),$$

where the $f_i$ are smooth but $r$ is non-smooth.

Examples: L1-regularization, bound constraints.

Proximal-gradient methods use iterations of the form

$$x^{k+1} = \text{prox}_{\alpha_k} \left[ x^k - \frac{\alpha_k}{n} \sum_{i=1}^{n} f'_i(x^k) \right],$$

and achieve the same rates as methods for smooth optimization.

Proximal-gradient variants of SAG[A]/MISO/SDCA/SVRG have been developed:

- Mairal [2013], Defazio et al. [2014], Xiao & Zhang [2014].

There are also combinations of these methods with ADMM:

- Suzuki [2014], Zhong & Kwok [2014].
Several Nesterov-like accelerated variants have been developed:

- SDCA [Shalev-Schwartz & Zhang, 2013, Shalev-Schwartz & Zhang, 2014].
- SVRG [Nitanda, 2014].
- Primal-Dual Coordinate Descent [Zhang & Xiao, 2014].
- All methods [Lin et al., 2015].
- RPDG [Lan, 2015].
- Catalyst [Lin et al., 2016].

Reduces complexity from $O((n + \kappa) \log(1/\epsilon))$ to $O(\sqrt{n\kappa} \log(1/\epsilon))$. 
Several Nesterov-like accelerated variants have been developed:

- SDCA [Shalev-Schwartz & Zhang, 2013, Shalev-Schwartz & Zhang, 2014].
- SVRG [Nitanda, 2014].
- Primal-Dual Coordinate Descent [Zhang & Xiao, 2014].
- All methods [Lin et al., 2015].
- RPDG [Lan, 2015].
- Catalyst [Lin et al., 2016].

Reduces complexity from $O((n + \kappa)\log(1/\epsilon))$ to $O(\sqrt{n\kappa}\log(1/\epsilon))$.

There also exist coordinate-wise and Newton-like variants:

- Konečnỳ et al. [2014], Sohl-Dickstein et al. [2014].
Consider case where each example has Lipschitz constant $L_i$. 

Justification: prefers gradients that change quickly. 

In practice: combine with line-search for adaptive sampling. (see paper/code for details)
Consider case where each example has Lipschitz constant $L_i$.

Non-uniform sampling proportional to $L_i$ in SVRG achieves

$$
\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(\bar{L}, \bar{L})[f(x^s) - f(x^*)],
$$

where $\bar{L} = \frac{1}{n} \sum_{i=1}^{n} L_i$.

Justification: prefers gradients that change quickly.
Consider case where each example has Lipschitz constant $L_i$.

Non-uniform sampling proportional to $L_i$ in SVRG achieves

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(\bar{L}, \bar{L})[f(x^s) - f(x^*)],$$

where $\bar{L} = \frac{1}{n} \sum_{i=1}^{n} L_i$.

Justification: prefers gradients that change quickly.

In practice: combine with line-search for adaptive sampling.

(see paper/code for details)
SAG with Non-Uniform Sampling

- protein \((n = 14571, p = 74)\) and sido \((n = 12678, p = 4932)\)

- Datasets where SAG had the worst relative performance.
protein \((n = 145751, \ p = 74)\) and sido \((n = 12678, \ p = 4932)\)

- Lipschitz sampling helps a lot.
CRF performance for optical-character and named-entity recognition.
Consider a truly-stochastic optimization problem,

$$\arg\min_x \mathbb{E}[f_i(x)].$$
Consider a truly-stochastic optimization problem,

\[ \arg\min_x \mathbb{E}[f_i(x)]. \]

Two classic regimes:

- Empirical risk minimization (ERM): optimize exactly over set of \( n \) samples.
- Stochastic gradient: apply \( n \) stochastic gradient iterations.

Recent alternative views suggest you can improve constants using:
- Growing batch sizes [Byrd et al., 2012].
- Re-visiting examples with SVRG [Babanezhad et al., 2015].
- Streaming SVRG [Frostig et al., 2015].
Consider a truly-stochastic optimization problem,

$$\arg\min_x \mathbb{E}[f_i(x)].$$

Two classic regimes:
- Empirical risk minimization (ERM): optimize exactly over set of $n$ samples.
- Stochastic gradient: apply $n$ stochastic gradient iterations.

Classic view: the above two approaches have $O(1/n)$ error.
- So ERM and fast stochastic gradient methods don’t help generalization?
Consider a truly-stochastic optimization problem,

$$\arg\min_x \mathbb{E}[f_i(x)].$$

Two classic regimes:
- Empirical risk minimization (ERM): optimize exactly over set of $n$ samples.
- Stochastic gradient: apply $n$ stochastic gradient iterations.

Classic view: the above two approaches have $O(1/n)$ error.
- So ERM and fast stochastic gradient methods don’t help generalization?

Classic view disagrees with practice: multiple passes usually helps.

Recent alternative views suggest you can improve constants using:
Consider a truly-stochastic optimization problem,

$$\arg\min_x \mathbb{E}[f_i(x)].$$

Two classic regimes:
- Empirical risk minimization (ERM): optimize exactly over set of \(n\) samples.
- Stochastic gradient: apply \(n\) stochastic gradient iterations.

Classic view: the above two approaches have \(O(1/n)\) error.
- So ERM and fast stochastic gradient methods don’t help generalization?

Classic view disagrees with practice: multiple passes usually helps.

Recent alternative views suggest you can improve constants using:
- Growing batch sizes [Byrd et al., 2012].
- Re-visiting examples with SVRG [Babanezhad et al., 2015].
- Streaming SVRG [Frostig et al., 2015].
Recent work on linearly-convergent stochastic gradient methods.
Recent work on linearly-convergent stochastic gradient methods.

SVRG is the only method without a memory requirement.
Recent work on linearly-convergent stochastic gradient methods.

SVRG is the only method without a memory requirement.

We give SVRG variants that reduce number of gradients.
Recent work on linearly-convergent stochastic gradient methods.

SVRG is the only method without a memory requirement.

We give SVRG variants that reduce number of gradients.

Speedups via regularization, acceleration, non-uniform sampling.
Recent work on linearly-convergent stochastic gradient methods.
SVRG is the only method without a memory requirement.
We give SVRG variants that reduce number of gradients.
Speedups via regularization, acceleration, non-uniform sampling.
Strong-convexity can relaxed:
  - Gong & Ye [2014], Garber & Hazan [2016], Karimi et al. [2016], Reddi et al. [2016]
Thank you for the invitation.