SVAN 2016 Mini-Course
Stochastic Convex Optimization Methods in Machine Learning

Mark Schmidt
University of British Columbia, May 2016

www.cs.ubc.ca/~schmidtm/SVAN16
Big-N Problems

- We can write our standard regularized optimization problem as

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)
\]

\[
\text{data fitting term} + \text{regularizer}
\]
We can write our standard regularized optimization problem as

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

data fitting term + regularizer

Gradient methods are effective when $d$ is very large.

What if number of training examples $n$ is very large?
  * E.g., ImageNet has more than 14 million annotated images.
Stochastic vs. Deterministic Gradient Methods

We consider minimizing \( f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \).
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]:

  \[
  x^{t+1} = x^t - \alpha_t \nabla f(x^t) = x^t - \frac{\alpha_t}{n} \sum_{i=1}^{n} \nabla f_i(x^t).
  \]

  - Iteration cost is **linear in** \( n \).
  - Convergence with constant \( \alpha_t \) or line-search.

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

  Random selection of \( i_t \) from \( \{1, 2, \ldots, n\} \).

  \[
  x^{t+1} = x^t - \alpha_t \nabla f_{i_t}(x^t).
  \]

  - Direction is an unbiased estimate of true gradient, \( ES[f'_{i_t}(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^t) = \nabla f(x^t) \).
  - Iteration cost is independent of \( n \).
  - Convergence requires \( \alpha_t \to 0 \).
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]:

  $$x^{t+1} = x^t - \alpha_t \nabla f(x^t) = x^t - \frac{\alpha_t}{n} \sum_{i=1}^{n} \nabla f_i(x^t).$$

  - Iteration cost is **linear in $n$**.
  - Convergence with constant $\alpha_t$ or line-search.

- **Stochastic** gradient method [Robbins & Monro, 1951]:
  - Random selection of $i_t$ from $\{1, 2, \ldots, n\}$.

  $$x^{t+1} = x^t - \alpha_t \nabla f_{i_t}(x^t).$$
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]:

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t) = x^t - \frac{\alpha_t}{n} \sum_{i=1}^{n} \nabla f_i(x^t).$$

- Iteration cost is linear in $n$.
- Convergence with constant $\alpha_t$ or line-search.

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

- Random selection of $i_t$ from $\{1, 2, \ldots, n\}$.

$$x^{t+1} = x^t - \alpha_t \nabla f_{i_t}(x^t).$$

- Direction is an unbiased estimate of true gradient,

$$\mathbb{E}[f'_{i_t}(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x).$$

- Iteration cost is independent of $n$. 
**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]:
  $$x^{t+1} = x^t - \alpha_t \nabla f(x^t) = x^t - \frac{\alpha_t}{n} \sum_{i=1}^{n} \nabla f_i(x^t).$$

  - Iteration cost is **linear in** $n$.
  - Convergence with constant $\alpha_t$ or line-search.

- **Stochastic** gradient method [Robbins & Monro, 1951]:
  - Random selection of $i_t$ from $\{1, 2, \ldots, n\}$.
  $$x^{t+1} = x^t - \alpha_t \nabla f_{i_t}(x^t).$$

  - Direction is an unbiased estimate of true gradient,
  $$\mathbb{E}[f'_{i_t}(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x).$$

  - Iteration cost is **independent of** $n$.
  - Convergence requires $\alpha_t \to 0$. 
We consider minimizing $f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$.

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

**Stochastic** gradient method [Robbins & Monro, 1951]:
Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are \( n \) times faster, but how many iterations?
Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are \( n \) times faster, but how many iterations?

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>( O(1/\sqrt{\epsilon}) )</td>
<td>( O(1/\epsilon^2) )</td>
</tr>
<tr>
<td>Strongly Convex</td>
<td>( O(\log(1/\epsilon)) )</td>
<td>( O(1/\epsilon) )</td>
</tr>
</tbody>
</table>

Stochastic has low iteration cost but slow convergence rate. Sublinear rate even in strongly-convex case. Bounds are unimprovable if only unbiased gradient available. Nesterov acceleration and momentum do not improve rate.
Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are \( n \) times faster, but how many iterations?

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>( O(1/\sqrt{\epsilon}) )</td>
<td>( O(1/\epsilon^2) )</td>
</tr>
<tr>
<td>Strongly</td>
<td>( O(\log(1/\epsilon)) )</td>
<td>( O(1/\epsilon) )</td>
</tr>
</tbody>
</table>

- Stochastic has low iteration cost but slow convergence rate.
  - Sublinear rate even in strongly-convex case.
  - Bounds are unimprovable if only unbiased gradient available.
Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are $n$ times faster, but how many iterations?

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\epsilon^2)$</td>
</tr>
<tr>
<td>Strongly</td>
<td>$O(\log(1/\epsilon))$</td>
<td>$O(1/\epsilon)$</td>
</tr>
</tbody>
</table>

- Stochastic has low iteration cost but slow convergence rate.
  - Sublinear rate even in strongly-convex case.
  - Bounds are unimprovable if only unbiased gradient available.
- Nesterov acceleration and momentum do not improve rate
Stochastic vs. Deterministic Convergence Rates

Plot of convergence rates in strongly-convex case:

\[
\begin{align*}
\text{time} & \\
\log(\text{excess cost}) & \\
\text{stochastic} & \\
\text{deterministic} &
\end{align*}
\]

Stochastic will be superior for low-accuracy/time situations.
Stochastic vs. Deterministic for Non-Smooth

The story changes for non-smooth problems.

Consider the binary support vector machine objective:

\[ f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\} + \frac{\lambda}{2} \|w\|^2. \]
The story changes for non-smooth problems. Consider the binary support vector machine objective:

\[ f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\} + \frac{\lambda}{2} \|w\|^2. \]

Rates for subgradient methods for non-smooth objectives:

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>(O(1/\epsilon^2))</td>
<td>(O(1/\epsilon^2))</td>
</tr>
<tr>
<td>Strongly</td>
<td>(O(1/\epsilon))</td>
<td>(O(1/\epsilon))</td>
</tr>
</tbody>
</table>

Other black-box methods (cutting plane) are not faster.
Stochastic vs. Deterministic for Non-Smooth

- The story changes for non-smooth problems.
- Consider the binary support vector machine objective:

\[ f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\} + \frac{\lambda}{2} \|w\|^2. \]

- Rates for subgradient methods for non-smooth objectives:

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>(O(1/\epsilon^2))</td>
<td>(O(1/\epsilon^2))</td>
</tr>
<tr>
<td>Strongly</td>
<td>(O(1/\epsilon))</td>
<td>(O(1/\epsilon))</td>
</tr>
</tbody>
</table>

- Other black-box methods (cutting plane) are not faster.
- For non-smooth problems:
  - Deterministic methods are not faster than stochastic method.
  - So use stochastic (iterations are \(n\) times faster).
Sub-Gradients and Sub-Differentials

Recall that for \textit{differentiable} convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T(y - x), \forall x, y.$$ 

A vector $d$ is a *subgradient* of a convex function $f$ at $x$ if

$$f(y) \geq f(x) + d^T(y - x), \forall y.$$
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
**Sub-Gradients and Sub-Differentials**

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$  

A vector $d$ is a *subgradient* of a convex function $f$ at $x$ if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a *subgradient* of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

\[ f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y. \]

A vector \( d \) is a subgradient of a convex function \( f \) at \( x \) if

\[ f(y) \geq f(x) + d^T (y - x), \forall y. \]
Sub-Gradients and Sub-Differentials

Recall that for differentiable convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$  

A vector $d$ is a subgradient of a convex function $f$ at $x$ if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$  

- At differentiable $x$:
  - Only subgradient is $\nabla f(x)$.

- At non-differentiable $x$:
  - We have a set of subgradients.
  - Called the sub-differential, $\partial f(x)$.
  - Sub-differential is always non-empty for (almost) all convex functions.

- Note that $0 \in \partial f(x)$ iff $x$ is a global minimum (generalizes $\nabla f(x) = 0$).
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

\[ \partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases} \]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

• Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0
\end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

\[ \partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0
\end{cases} \]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

\[ \partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases} \]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:

\[
\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases}
\]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:

\[
\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases}
\]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

Sub-differential of **absolute value** function:

$$
\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0
\end{cases}
$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:

\[
\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases}
\]

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

\[ \partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0 
\end{cases} \]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:

\[
\partial |x| = \begin{cases}
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0
\end{cases}
\]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:
  \[
  \partial |x| = \begin{cases} 
  1 & x > 0 \\
  -1 & x < 0 \\
  [-1, 1] & x = 0 
  \end{cases}
  \]
  (sign of the variable if non-zero, anything in \([-1, 1]\) at 0)

- Sub-differential of sum of convex \(f_1\) and \(f_2\):
  \[
  \partial (f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).
  \]
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:
  \[
  \partial |x| = \begin{cases} 
  1 & x > 0 \\
  -1 & x < 0 \\
  [-1, 1] & x = 0
  \end{cases}
  \]
  (sign of the variable if non-zero, anything in \([-1, 1]\) at 0)

- Sub-differential of **sum** of convex \(f_1\) and \(f_2\):
  \[
  \partial (f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).
  \]

- Sub-differential of **max** of convex \(f_1\) and \(f_2\):
  \[
  \partial \max\{f_1(x), f_2(x)\} =
  \]
Sub-Differential of Absolute Value and Max Functions

- Sub-differential of **absolute value** function:

\[
\partial |x| = \begin{cases} 
1 & x > 0 \\
-1 & x < 0 \\
[-1, 1] & x = 0
\end{cases}
\]

(sign of the variable if non-zero, anything in \([-1, 1]\) at 0)

- Sub-differential of **sum** of convex \(f_1\) and \(f_2\):

\[
\partial (f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).
\]

- Sub-differential of **max** of convex \(f_1\) and \(f_2\):

\[
\partial \max\{f_1(x), f_2(x)\} = \begin{cases} 
\nabla f_1(x) & f_1(x) > f_2(x) \\
\nabla f_2(x) & f_2(x) > f_1(x) \\
\theta \nabla f_1(x) + (1 - \theta) \nabla f_2(x) & f_1(x) = f_2(x)
\end{cases}
\]

(any “convex combination” of the gradients of the argmax)
Subgradient Method

- The basic subgradient method:

\[ x^{t+1} = x^t - \alpha_t g_t, \]

for some \( g_t \in \partial f(x^t) \).
Subgradient Method

- The basic subgradient method:
  \[ x^{t+1} = x^t - \alpha_t g_t, \]
  for some \( g_t \in \partial f(x^t) \).
- Unfortunately, may increase the objective even for small \( \alpha_t \).
- But, distance to solution decreases:
  - \( \|x^{t+1} - x^*\| < \|x^t - x^*\| \) for small enough \( \alpha \).
Subgradient Method

- The basic subgradient method:

\[ x^{t+1} = x^t - \alpha_t g_t, \]

for some \( g_t \in \partial f(x^t) \).

- Unfortunately, may increase the objective even for small \( \alpha_t \).

- But, distance to solution decreases:
  
  \[ \|x^{t+1} - x^*\| < \|x^t - x^*\| \] for small enough \( \alpha \).
Subgradient Method

- The basic subgradient method:

\[ x^{t+1} = x^t - \alpha_t g_t, \]

for some \( g_t \in \partial f(x^t) \).

- Unfortunately, may increase the objective even for small \( \alpha_t \).

- But, distance to solution decreases:
  - \( ||x^{t+1} - x^*|| < ||x^t - x^*|| \) for small enough \( \alpha \).
Subgradient Method

- The basic subgradient method:

\[ x^{t+1} = x^t - \alpha_t g_t, \]

for some \( g_t \in \partial f(x^t) \).

- Unfortunately, may increase the objective even for small \( \alpha_t \).

- But, distance to solution decreases:
  - \( ||x^{t+1} - x^*|| < ||x^t - x^*|| \) for small enough \( \alpha \).
The basic subgradient method:

\[ x^{t+1} = x^t - \alpha_t g_t, \]

for some \( g_t \in \partial f(x^t) \).

Unfortunately, may increase the objective even for small \( \alpha_t \).

But, distance to solution decreases:

\[ \|x^{t+1} - x^*\| < \|x^t - x^*\| \]

for small enough \( \alpha \).
Strong-Convexity Inequalities for Non-Differentiable $f$

A “first-order” relationship between subgradient and strong-convexity:
- If $f$ is $\mu$-strongly convex then for all $x$ and $y$ we have

$$f(y) \geq f(x) + f'(y)^T(y - x) + \frac{\mu}{2}\|y - x\|^2,$$

for $f'(y) \in \partial f(x)$.

The first-order definition of strong-convexity, but with subgradient replacing gradient.
A “first-order” relationship between subgradient and strong-convexity:

If $f$ is $\mu$-strongly convex then for all $x$ and $y$ we have

$$f(y) \geq f(x) + f'(y)^T(y - x) + \frac{\mu}{2} \|y - x\|^2,$$

for $f'(y) \in \partial f(x)$.

The first-order definition of strong-convexity, but with subgradient replacing gradient.

Reversing $y$ and $x$ we can write

$$f(x) \geq f(y) + f'(x)^T(x - y) + \frac{\mu}{2} \|x - y\|^2,$$

for $f'(x) \in \partial f(x)$.

Adding the above together gives

$$(f'(y) - f'(x)^T(y - x)) \geq \mu \|y - x\|^2.$$
Stochastic Subgradient Method

- The basic stochastic subgradient method:

\[ x^{t+1} = x^t - \alpha g_{i_t}, \]

for some \( g_{i_t} \in \partial f_{i_t}(x^t) \) for some random \( i_t \in \{1, 2, \ldots, n\} \).
The basic **stochastic** subgradient method:

\[ x^{t+1} = x^t - \alpha g_{i_t}, \]

for some \( g_{i_t} \in \partial f_{i_t}(x^t) \) for some random \( i_t \in \{1, 2, \ldots, n\} \).

- Stochastic subgradient is \( n \) times faster with similar convergence properties.
- We’ll consider it under the standard assumptions that
  - \( f \) is \( \mu \)-strongly-convex:
  - \( \mathbb{E}[\|g_t\|^2] \leq B^2 \) (finite variance and bounded subgradients).
Since function value may not decrease, we analyze distance to $x^*$:

$$
\|x^t - x^*\|^2 = \|(x^{t-1} - \alpha_t g_i) - x^*\|^2
= \|(x^{t-1} - x^*) - \alpha_t g_i\|^2
= \|x^{t-1} - x^*\|^2 - 2\alpha_t g_i^T (x^{t-1} - x^*) + \alpha_t^2 \|g_i\|^2.
$$
Since function value may not decrease, we analyze distance to $x^*$:

$$\|x^t - x^*\|^2 = \|(x^{t-1} - \alpha_t g_{it}) - x^*\|^2$$
$$= \|(x^{t-1} - x^*) - \alpha_t g_{it}\|^2$$
$$= \|x^{t-1} - x^*\|^2 - 2\alpha_t g_{it}^T(x^{t-1} - x^*) + \alpha_t^2 \|g_{it}\|^2.$$

Many analyses of distance to $x^*$ start this way.

First term is what we want, we need to bound the second/third terms.
Convergence Rate of Stochastic Subgradient

- Expansion of distance:

\[ \|x^t - x^*\|^2 = \|x^{t-1} - x^*\|^2 - 2\alpha_t g^T_i (x^{t-1} - x^*) + \alpha_t^2 \|g^i_t\|^2. \]
Convergence Rate of Stochastic Subgradient

- Expansion of distance:
  \[ \|x^t - x^*\|^2 = \|x^{t-1} - x^*\|^2 - 2\alpha_t g_i^T (x^{t-1} - x^*) + \alpha_t^2 \|g_i\|^2. \]

- Take expectation with respect to \(i_t\):
  \[
  \mathbb{E}[\|x^t - x^*\|^2] = \mathbb{E}[\|x^{t-1} - x^*\|^2] - 2\alpha_t \mathbb{E}[g_i^T (x^{t-1} - x^*)] + \alpha_t^2 \mathbb{E}[\|g_i\|^2]
  \]
  \[
  \|x^{t-1} - x^*\|^2 - 2\alpha_t \mathbb{E}[g_i^T (x^{t-1} - x^*)] + \alpha_t^2 \mathbb{E}[\|g_i\|^2]
  \]
  \[
  \leq \|x^{t-1} - x^*\|^2 - 2\alpha_t g_i^T (x^{t-1} - x^*) + \alpha_t^2 B^2.
  \]
Convergence Rate of Stochastic Subgradient

- Expansion of distance:
  \[ \| x^t - x^* \|^2 = \| x^{t-1} - x^* \|^2 - 2\alpha_t g_{i_t}^T (x^{t-1} - x^*) + \alpha_t^2 \| g_{i_t} \|^2. \]

- Take expectation with respect to \( i_t \):
  \[
  \mathbb{E}[\| x^t - x^* \|^2] = \mathbb{E}[\| x^{t-1} - x^* \|^2] - 2\alpha_t \mathbb{E}[g_{i_t}^T (x^{t-1} - x^*)] + \alpha_t^2 \mathbb{E}[\| g_{i_t} \|^2] \\
  \leq \| x^{t-1} - x^* \|^2 - 2\alpha_t \mathbb{E}[g_{i_t}^T (x^{t-1} - x^*)] + \alpha_t^2 \mathbb{E}[\| g_{i_t} \|^2] \\
  \leq \| x^{t-1} - x^* \|^2 - 2\alpha_t g_{i_t}^T (x^{t-1} - x^*) + \alpha_t^2 B^2.
  \]

- Using strong-convexity inequality,
  \[(g_t - 0)^T (x^{t-1} - x^*) \geq \mu \| y - x \|^2,
  \]
gives
  \[
  \mathbb{E}[\| x^t - x^* \|^2] \leq \| x^{t-1} - x^* \|^2 - 2\alpha_t \mu \| x^{t-1} - x^* \|^2 + \alpha_t^2 B^2 \\
  = (1 - 2\alpha_t \mu) \| x^{t-1} - x^* \|^2 + \alpha_t^2 B^2.
  \]
Stochastic Gradient with Constant Step Size

- Our bound on expected distance:
  \[ \mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha_t \mu)\|x^{t-1} - x^*\|^2 + \alpha_t^2 B^2. \]

- If \( \alpha_t \) is small enough, shows distance to solution decreases.
Stochastic Gradient with Constant Step Size

- Our bound on expected distance:
  \[ \mathbb{E}[\|x_t - x^*\|^2] \leq (1 - 2\alpha_t \mu)\|x_{t-1} - x^*\|^2 + \alpha_t^2 B^2. \]

- If \( \alpha_t \) is small enough, shows distance to solution decreases.

- Taking full expectation and applying recursively with constant \( \alpha_t = \alpha \) gives:
  \[ \mathbb{E}[\|x_t - x^*\|^2] \leq (1 - 2\alpha \mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}, \]
  after some of math (last term comes from bounding a geometric series).

- First term looks like linear convergence, but second term does not go to zero.
Stochastic Gradient with Constant Step Size

- Our bound on expected distance:
  \[
  \mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}.
  \]

- First term looks like linear convergence, but second term does not go to zero.
Stochastic Gradient with Constant Step Size

- Our bound on expected distance:
  \[ \mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}. \]

- First term looks like linear convergence, but second term does not go to zero.
Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

\[ \mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha \mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}. \]

- First term looks like **linear convergence**, but second term does not go to zero.
**Stochastic Gradient with Constant Step Size**

- Our bound on expected distance:

  \[ \mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}. \]

- First term looks like *linear convergence*, but second term does not go to zero.
To get convergence, we need a decreasing step size.

- Region that we converge to shrinks over time.
- But it can’t shrink too quickly or we may never reach $x^*$.
Stochastic Subgradient Convergence Rate of SSG Practical Subgradient Methods Stochastic Average Gradient Infinite Data Sets

Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - Region that we converge to shrinks over time.
  - But it can’t shrink too quickly or we may never reach $x^*$.
  - Classic approach is to choose $\alpha_t$ such that

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty,$$

  which suggests setting $\alpha_t = O(1/t)$. 
Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - Region that we converge to shrinks over time.
  - But it can’t shrink too quickly or we may never reach \( x^* \).
  - Classic approach is to choose \( \alpha_t \) such that

\[
\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty,
\]

which suggests setting \( \alpha_t = O(1/t) \).
- We can obtain convergence rates with decreasing steps:
  - If \( \alpha_t = \frac{1}{\mu t} \) we can show

\[
\mathbb{E}[f(\bar{x}^t) - f(x^*)] = O(\log(t)/t) \quad \text{(non-smooth } f) \\
= O(1/t) \quad \text{(smooth } f)
\]

for the average iteration \( \bar{x}^t = \frac{1}{k} \sum_{k=1}^{T} x_{k-1} \).
- Note that \( O(1/t) \) error implies \( O(1/\epsilon) \) iterations required.
(pause)
What is the best subgradient?

- We analyzed the subgradient method,

\[ x^{t+1} = x^t - \alpha_t g_t, \text{ where } g_t \in \partial f(x^t), \]

under any choice of subgradient.
What is the best subgradient?

- We analyzed the subgradient method,

\[ x^{t+1} = x^t - \alpha_t g_t, \text{ where } g_t \in \partial f(x^t), \]

under any choice of subgradient.

- But what is the “best” subgradient to use?
  - Convex functions have directional derivatives everywhere.
  - Direction \(-z^t\) that minimizes directional derivative is minimum-norm subgradient,

\[ z^t = \arg\min_{z \in \partial f(x^t)} ||z|| \]

- This is the steepest descent direction for non-smooth convex optimization problems.
What is the best subgradient?

- We analyzed the subgradient method,

\[ x^{t+1} = x^t - \alpha_t g_t, \quad \text{where} \quad g_t \in \partial f(x^t), \]

under any choice of subgradient.

- But what is the “best” subgradient to use?

  - Convex functions have directional derivatives everywhere.
  - Direction \(-z^t\) that minimizes directional derivative is minimum-norm subgradient,

\[ z^t = \arg\min_{z \in \partial f(x^t)} ||z|| \]

  - This is the steepest descent direction for non-smooth convex optimization problems.
  - You can compute this for L1-regularization, but not many other problems.
  - Basis for best L1-regularization methods, combined (carefully) with Newton.
Stochastic Subgradient with Sparse Features

- For many datasets, our feature vectors \( x_i \) are very sparse:

<table>
<thead>
<tr>
<th>&quot;CPSC&quot;</th>
<th>&quot;Expedia&quot;</th>
<th>&quot;vicodin&quot;</th>
<th>&lt;recipient name&gt;</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

- Consider case where \( d \) is huge but each row \( x_i \) has at most \( k \) non-zeroes:
  - The \( O(d) \) cost of stochastic subgradient might be too high.
  - We can often modify stochastic subgradient to have \( O(k) \) cost.
Digression: Operations on Sparse Vectors

- Consider a vector \( g \in \mathbb{R}^d \) with at most \( k \) non-zeroes:

\[
g^T = [0 \ 0 \ 0 \ 1 \ 2 \ 0 \ -0.5 \ 0 \ 0 \ 0].
\]

- If \( k \ll d \), we can store the vector using \( O(k) \) storage instead of \( O(d) \):
Digression: Operations on Sparse Vectors

- Consider a vector $g \in \mathbb{R}^d$ with at most $k$ non-zeroes:
  \[
g^T = [0 \ 0 \ 0 \ 1 \ 2 \ 0 \ -0.5 \ 0 \ 0 \ 0].
  \]

- If $k << d$, we can store the vector using $O(k)$ storage instead of $O(d)$:
  - Store the non-zero values:
    \[
g_{\text{value}}^T = [1 \ 2 \ -0.5].
    \]
  - Store a pointer to where the non-zero values go:
    \[
g_{\text{point}}^T = [4 \ 5 \ 7].
    \]
Digression: Operations on Sparse Vectors

- Consider a vector $g \in \mathbb{R}^d$ with at most $k$ non-zeros:
  
  $$g^T = [0 \ 0 \ 0 \ 1 \ 2 \ 0 \ -0.5 \ 0 \ 0 \ 0].$$

- If $k \ll d$, we can store the vector using $O(k)$ storage instead of $O(d)$:
  
  - Store the non-zero values:
    
    $$g^T_{value} = [1 \ 2 \ -0.5].$$

  - Store a pointer to where the non-zero values go:
    
    $$g^T_{point} = [4 \ 5 \ 7].$$

- With this representation, we can do standard vector operations in $O(k)$:
  
  - Compute $\alpha g$ in $O(k)$ by computing $\alpha g_{value}$.
  - For dense $w$, set $w = (w - g)$ in $O(k)$ by subtracting $g_{value}$ from $w$ at positions $g_{point}$.
Stochastic Subgradient with Sparse Features

Consider optimizing the hinge-loss,

$$
\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\},
$$

when \(d\) is huge but each row has at most \(k\) non-zeroes.
Stochastic Subgradient with Sparse Features

- Consider optimizing the hinge-loss,

\[
\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\},
\]

when \(d\) is huge but each row has at most \(k\) non-zeroes.

- A stochastic subgradient method could use

\[
w^{t+1} = w^t - \alpha_t g_i, \quad \text{where} \quad g_i = \begin{cases} 
- y_i x_i & \text{if } 1 - y_i(w^T x_i) > 0 \\
0 & \text{otherwise}
\end{cases}
\]
Stochastic Subgradient with Sparse Features

- Consider optimizing the hinge-loss,

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i (w^T x_i)\},$$

when \(d\) is huge but each row has at most \(k\) non-zeroes.

- A stochastic subgradient method could use

$$w^{t+1} = w^t - \alpha_t g_t,$$

where \(g_i = \begin{cases} -y_i x_i & \text{if } 1 - y_i (w^T x_i) > 0 \\ 0 & \text{otherwise} \end{cases}\)

- Notice that \(g_i\) has at most \(k\) non-zeroes:
  - Computing \(\alpha_t g_i\) costs \(O(k)\): multiply \(\alpha_t\) by non-zeroes.
  - Computing \(w^t - \alpha_t g_i\) costs \(O(k)\): subtract non-zeroes.

- So stochastic subgradient is fast if \(k\) is small even if \(d\) is large.
Stochastic Subgradient with Sparse Features

Consider the L2-regularized hinge-loss in the same setting,

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x_i)\} + \frac{\lambda}{2} \|w\|^2,$$

using a stochastic subgradient method,

$$w^{t+1} = w^t - \alpha_t g_{it} - \alpha_t \lambda w^t,$$

where $g_{it}$ is same as before.
Stochastic Subgradient with Sparse Features

- Consider the **L2-regularized** hinge-loss in the same setting,

\[
\text{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i (w^T x_i)\} + \frac{\lambda}{2} \|w\|^2,
\]

using a stochastic subgradient method,

\[
w^{t+1} = w^t - \alpha_t g_{it} - \alpha_t \lambda w^t,
\]

where \(g_{it}\) is same as before.

- While \(g_{it}\) has at most \(k\) non-zeros, \(w^t\) could have \(d\) non-zeroes:
  - So adding L2-regularization increases cost from \(O(k)\) to \(O(d)\)?
Stochastic Subgradient with Sparse Features

Consider the L2-regularized hinge-loss in the same setting,

\[
\text{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i (w^T x_i)\} + \frac{\lambda}{2} \|w\|^2,
\]

using a stochastic subgradient method,

\[
w^{t+1} = w^t - \alpha_t g_{it} - \alpha_t \lambda w^t,
\]

where \(g_{it}\) is same as before.

While \(g_{it}\) has at most \(k\) non-zeroes, \(w^t\) could have \(d\) non-zeroes:

- So adding L2-regularization increases cost from \(O(k)\) to \(O(d)\)?

To use L2-regularization and keep \(O(k)\) cost, re-write iteration as

\[
w^{t+1} = w^t - \alpha_t g_{it} - \alpha_t \lambda w^t
\]

\[
= (1 - \alpha_t \lambda)w^t - \alpha_t g_{it}.
\]

changes scale of \(w^t\) sparse update
Stochastic Subgradient with Sparse Features

Let’s write the update as two steps

\[ w^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)w^t, \quad w^{t+1} = w^{t+\frac{1}{2}} - \alpha_t g_{i_t}. \]
Stochastic Subgradient with Sparse Features

- Let’s write the update as two steps
  \[ w^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)w^t, \quad w^{t+1} = w^{t+\frac{1}{2}} - \alpha_t g_i. \]

- We can implement both steps in \( O(k) \) if we re-parameterize as
  \[ w^t = \beta^t v^t, \]

for some scalar \( \beta^t \) and vector \( v^t \).
Stochastic Subgradient with Sparse Features

Let’s write the update as two steps

\[ w^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)w^t, \quad w^{t+1} = w^{t+\frac{1}{2}} - \alpha_t g_{i_t}. \]

We can implement both steps in \(O(k)\) if we re-parameterize as

\[ w^t = \beta^t v^t, \]

for some scalar \(\beta^t\) and vector \(v^t\).

For the first step we need

\[ \beta^{t+\frac{1}{2}} v^{t+\frac{1}{2}} = (1 - \alpha_t \lambda) \beta^t v^t, \]

which we can satisfy in \(O(1)\) using \(\beta^{t+\frac{1}{2}} = (1 - \alpha_t \lambda) \beta^t\) and \(v^{t+\frac{1}{2}} = v^t\).
Stochastic Subgradient with Sparse Features

- Let’s write the update as two steps
  \[ w^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)w^t, \quad w^{t+1} = w^{t+\frac{1}{2}} - \alpha_t g_i. \]

- We can implement both steps in \( O(k) \) if we re-parameterize as
  \[ w^t = \beta^t v^t, \]
  for some scalar \( \beta^t \) and vector \( v^t \).

- For the first step we need
  \[ \beta^{t+\frac{1}{2}} v^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)\beta^t v^t, \]
  which we can satisfy in \( O(1) \) using \( \beta^{t+\frac{1}{2}} = (1 - \alpha_t \lambda)\beta^t \) and \( v^{t+\frac{1}{2}} = v^t \).

- For the second step we need
  \[ \beta^{t+1} v^{t+1} = \beta^{t+\frac{1}{2}} v^{t+\frac{1}{2}} - \alpha_t g_i, \]
  which we can satisfy in \( O(k) \) using \( \beta^{t+1} = \beta^{t+\frac{1}{2}} \) and \( v^{t+1} = v^{t+\frac{1}{2}} - \frac{\alpha_t}{\beta^{t+\frac{1}{2}}} g_i \).
Stochastic Subgradient with Sparse Features

- So we can implement the subgradient method with L2-regularization, 

\[ w^{t+1} = w^t - \alpha_t g_t - \alpha_t \lambda w^t, \]

in \( O(k) \) by using the \( w^t = \beta^t v^t \) representation and the update 

\[ \beta^{t+1} = (1 - \alpha_t \lambda) \beta^t, \quad v^{t+1} = v^t - \frac{\alpha_t}{\beta^{t+1}} g_t. \]

assuming that computing \( g_t \) can be done in \( O(k) \) given \( \beta^t \) and \( v^t \).
Stochastic Subgradient with Sparse Features

- So we can implement the subgradient method with L2-regularization,

\[ w^{t+1} = w^t - \alpha_t g_i^t - \alpha_t \lambda w^t, \]

in \( O(k) \) by using the \( w^t = \beta^t v^t \) representation and the update

\[ \beta^{t+1} = (1 - \alpha_t \lambda) \beta^t, \quad v^{t+1} = v^t - \frac{\alpha_t}{\beta^{t+1}} g_i^t. \]

assuming that computing \( g_i^t \) can be done in \( O(k) \) given \( \beta^t \) and \( v^t \).

- There exists efficient sparse updates in other scenarios too:
  - Duchi & Singer [2009]: L1-regularization proximal operator ("lazy updates").
  - Xu [2010]: L2-regularization and iterate average \( \bar{w}^t \).
Stochastic Subgradient Methods in Practice

- Last time we argued that $\alpha_t$ must go to zero for convergence.
- Theory says using $\alpha_t = 1/\mu t$ and averaging is close to optimal:
Stochastic Subgradient Methods in Practice

- Last time we argued that $\alpha_t$ must go to zero for convergence.
- Theory says using $\alpha_t = 1/\mu t$ and averaging is close to optimal:
  - Except for some special cases, you should not do this.
  - Usually $\mu = O(1/n)$ or $O(1/\sqrt{n})$ so initial steps are huge.
  - Later steps are tiny: $1/t$ gets small very quickly.
  - Convergence rate slows dramatically if $\mu$ isn’t accurate.
  - No adaptation to “easier” problems than worst case.

Tricks that can improve theoretical and practical properties:

1. Use smaller initial step-sizes, that go to zero more slowly: $\alpha_t = \gamma/\sqrt{t}$ or $\alpha_t = \gamma$.
2. Take a (weighted) average of the iterations or gradients:
   $$\bar{x}_t = \frac{t}{\sum_{i=1}^t \omega_t z_t},$$
   where $\omega_t$ is weight at iteration $t$.

These tricks usually help, but tuning is often required: stochastic subgradient is not a black box.
Last time we argued that \( \alpha_t \) must go to zero for convergence.

Theory says using \( \alpha_t = 1/\mu t \) and averaging is close to optimal:

- Except for some special cases, you should not do this.
- Usually \( \mu = O(1/n) \) or \( O(1/\sqrt{n}) \) so initial steps are huge.
- Later steps are tiny: \( 1/t \) gets small very quickly.
- Convergence rate slows dramatically if \( \mu \) isn’t accurate.
- No adaptation to “easier” problems than worst case.

Tricks that can improve theoretical and practical properties:

1. Use smaller initial step-sizes, that go to zero more slowly:
   \[
   \alpha_t = \gamma/\sqrt{t} \quad \text{or} \quad \alpha_t = \gamma.
   \]

2. Take a (weighted) average of the iterations or gradients:
   \[
   \bar{x}^t = \sum_{i=1}^{t} \omega_t z^t,
   \]
   where \( \omega_t \) is weight at iteration \( t \).
Stochastic Subgradient Methods in Practice

- Last time we argued that $\alpha_t$ must go to zero for convergence.
- Theory says using $\alpha_t = 1/\mu t$ and averaging is close to optimal:
  - Except for some special cases, you should not do this.
  - Usually $\mu = O(1/n)$ or $O(1/\sqrt{n})$ so initial steps are huge.
  - Later steps are tiny: $1/t$ gets small very quickly.
  - Convergence rate slows dramatically if $\mu$ isn’t accurate.
  - No adaptation to “easier” problems than worst case.

- Tricks that can improve theoretical and practical properties:
  1. Use smaller initial step-sizes, that go to zero more slowly:
     $$\alpha_t = \gamma/\sqrt{t} \quad \text{or} \quad \alpha_t = \gamma.$$
  2. Take a (weighted) average of the iterations or gradients:
     $$\bar{x}^t = \sum_{i=1}^{t} \omega_t z^t,$$
     where $\omega_t$ is weight at iteration $t$.

- These tricks usually help, but tuning is often required:
  - stochastic subgradient is not a black box.
Speeding up Stochastic Subgradient Methods

Results that support using large steps and averaging:

- Averaging later iterations achieves $O(1/t)$ in non-smooth case.
- Gradient averaging improves constants in analysis.
- $\alpha_t = O(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = O(1/t)$. 
Speeding up Stochastic Subgradient Methods

Results that support using large steps and averaging:

- **Averaging later iterations** achieves $O(1/t)$ in non-smooth case.
- **Gradient averaging** improves constants in analysis.
- $\alpha_t = O(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = O(1/t)$.
- **Constant step size** ($\alpha_t = \alpha$) achieves linear rate to accuracy $O(\alpha)$.
- In smooth case, **iterate averaging is asymptotically optimal**:
  - Achieves same rate as optimal stochastic Newton method.
Stochastic Newton Methods?

- Should we use Nesterov/Newton-like stochastic methods?
  - These do not improve the $O(1/\epsilon)$ convergence rate.
Stochastic Newton Methods?

- Should we use Nesterov/Newton-like stochastic methods?
  - These do not improve the $O(1/\epsilon)$ convergence rate.
- But some positive results exist.
  - Improves performance at start or if noise is small.
  - Newton-like AdaGrad method,

$$x^{t+1} = x^t + \alpha D \nabla f_i(x^t), \quad \text{with } D_{jj} = \sqrt{\sum_{k=1}^{t} \|\nabla_j f_{i_k}(x^t)\|}.$$  

- improves regret but not optimization error.
- Two-phase Newton-like method achieves $O(1/\epsilon)$ without strong-convexity.
<table>
<thead>
<tr>
<th>Stochastic Subgradient</th>
<th>Convergence Rate of SSG</th>
<th>Practical Subgradient Methods</th>
<th>Stochastic Average Gradient</th>
<th>Infinite Data Sets</th>
</tr>
</thead>
</table>

(pause)
Recall our standard optimization framework,

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)
\]

- data fitting term  +  regularizer
Big-N Problems

- Recall our standard optimization framework,

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)
\]

- data fitting term + regularizer

- **Stochastic methods:**
  - \(O(1/t)\) convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
Big-N Problems

- Recall our standard optimization framework,

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)
\]

data fitting term + regularizer

- Stochastic methods:
  - \(O(1/t)\) convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.

- Deterministic methods:
  - \(O(\rho^t)\) convergence but requires \(N\) gradients per iteration.
  - The faster rate is possible because \(N\) is finite.
Recall our standard optimization framework,

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

\[ \text{data fitting term} \quad + \quad \text{regularizer} \]

- **Stochastic methods:**
  - \(O(1/t)\) convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.

- **Deterministic methods:**
  - \(O(\rho^t)\) convergence but requires \(N\) gradients per iteration.
  - The faster rate is possible because \(N\) is finite.

- For minimizing finite sums, can we design a better method?
Motivation for Hybrid Methods

Stochastic vs. deterministic methods
• Goal = best of both worlds: linear rate with $O(1)$ iteration cost
time $\log(\text{excess cost})$

- stochastic
- deterministic
Motivation for Hybrid Methods

Stochastic vs. deterministic methods

- Goal: best of both worlds: \( \text{linear rate with } O(1) \text{ iteration cost} \)
- Hybrid: \( \log(\text{excess cost}) \)
  - Stochastic
  - Deterministic
  - Hybrid

Graph showing the relationship between log(excess cost) and time for stochastic, deterministic, and hybrid methods.
Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
- The FG method uses all $N$ gradients,

$$\nabla f(x^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x^t).$$

- The SG method approximates it with 1 sample,

$$\nabla f_{i t}(x^t) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x^t).$$
Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
- The FG method uses all $N$ gradients,

$$\nabla f(x^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x^t).$$

- The SG method approximates it with 1 sample,

$$\nabla f_{i_t}(x^t) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x^t).$$

- A common variant is to use larger sample $\mathcal{B}^t$,

$$\frac{1}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} \nabla f_i(x^t) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x^t).$$
Approach 1: Batching

- The SG method with a sample $B^t$ uses iterations

$$x^{t+1} = x^t - \frac{\alpha^t}{|B^t|} \sum_{i \in B^t} f_i(x^t).$$

- For a fixed sample size $|B^t|$, the rate is sublinear.
Approach 1: Batching

- The SG method with a sample $B^t$ uses iterations
  \[
  x^{t+1} = x^t - \alpha^t \frac{1}{|B^t|} \sum_{i \in B^t} f_i(x^t).
  \]

- For a fixed sample size $|B^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|B^t|$ increases.**
Approach 1: Batching

- The SG method with a sample $B^t$ uses iterations

\[ x^{t+1} = x^t - \frac{\alpha^t}{|B^t|} \sum_{i \in B^t} f_i(x^t). \]

- For a fixed sample size $|B^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|B^t|$ increases.**
- Common to gradually increase the sample size $|B^t|$.

[Bertsekas & Tsitsiklis, 1996]
Approach 1: Batching

- The SG method with a sample $B^t$ uses iterations

$$x^{t+1} = x^t - \frac{\alpha^t}{|B^t|} \sum_{i \in B^t} f_i(x^t).$$

- For a fixed sample size $|B^t|$, the rate is sublinear.

- **Gradient error decreases as sample size** $|B^t|$ **increases**.

- Common to gradually increase the sample size $|B^t|$.

  [Bertsekas & Tsitsiklis, 1996]

- **We can choose** $|B^t|$ **to achieve a linear convergence rate**:
  - Early iterations are cheap like SG iterations.
  - Later iterations can use a Newton-like method.
Stochastic Average Gradient

- Growing $|B^t|$ eventually requires $O(N)$ iteration cost.
- Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?
Stochastic Average Gradient

- Growing $|B^t|$ eventually requires $O(N)$ iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - **YES!**
Stochastic Average Gradient

- Growing $|B_t|$ eventually requires $O(N)$ iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i_t$ from $\{1, 2, \ldots, N\}$ and compute $f'_i(x^t)$.
    
    $$x^{t+1} = x^t - \alpha^t \frac{N}{\sum_{i=1}^{N} \nabla f_i(x^t)}$$
Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires $O(N)$ iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - **YES!** The stochastic average gradient (SAG) algorithm:
    - Randomly select $i_t$ from $\{1, 2, \ldots, N\}$ and compute $f_{i_t}'(x^t)$.

\[
x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} \nabla f_i(x^t)
\]
Stochastic Average Gradient

- Growing $|B^t|$ eventually requires $O(N)$ iteration cost.
- **Can we have a rate of** $O(\rho^t)$ **with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select $i_t$ from $\{1, 2, \ldots, N\}$ and compute $f'_i(x^t)$.
    - $x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$
    - **Memory:** $y_i^t = \nabla f_i(x^t)$ from the last $t$ where $i$ was selected. [Le Roux et al., 2012]
Stochastic Average Gradient

- Growing $|B^t|$ eventually requires $O(N)$ iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - 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:** $y^t_i = \nabla f_i(x^t)$ from the last $t$ where $i$ was selected.
  - [Le Roux et al., 2012]
- **Stochastic** variant of increment average gradient (IAG).
  - [Blatt et al., 2007]
Stochastic Average Gradient

Growing $|B^t|$ eventually requires $O(N)$ iteration cost.

Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?

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:** $y^t_i = \nabla f_i(x^t)$ from the last $t$ where $i$ was selected.
  
  [Le Roux et al., 2012]

- **Stochastic** variant of increment average gradient (IAG).
  
  [Blatt et al., 2007]

- Assumes gradients of non-selected examples don’t change.
- Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$. 

Convergence Rate of SAG

- If each $f'_i$ is $L$–continuous and $f$ is strongly-convex, with $\alpha_t = 1/16L$ SAG has

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

where

$$C = [f(x^0) - f(x^*)] + \frac{4L}{N} \|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$
Stochastic Subgradient Convergence Rate of SSG Practical Subgradient Methods Stochastic Average Gradient Infinite Data Sets

**Convergence Rate of SAG**

- If each $f_i'$ is $L-$continuous and $f$ is strongly-convex, with $\alpha_t = 1/16L$ SAG has

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

where

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

- Linear convergence rate but only 1 gradient per iteration.
  - For well-conditioned problems, constant reduction per pass:
    $$\left(1 - \frac{1}{8N}\right)^N \leq \exp \left(-\frac{1}{8}\right) = 0.8825.$$
  - For ill-conditioned problems, almost same as deterministic method (but $N$ 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 $(1 - \min\{\mu 16L, 18N\})N = 0.88250$.

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: $O(\max\{N, L \mu \} \log(1/\epsilon))$. 
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 $\left( 1 - \sqrt{\mu L} \right) = 0.99761$. 
  - SAG ($N$ iterations) has rate $\left( 1 - \min \{ \frac{\mu}{16L}, \frac{1}{8N} \} \right) = 0.88250$. 
  - Fastest possible first-order method: $\left( \sqrt{L} - \sqrt{\mu} \right)^2 = 0.99048$. 

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 $\left( 1 - \sqrt{\frac{\mu}{L}} \right) = 0.99761$. 
Rate of Convergence Comparison

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


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 \((1 - \sqrt{\frac{\mu}{L}}) = 0.99761\).
- 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: $O\left(\frac{L}{\mu}(1/\epsilon)\right)$.
- Gradient: $O\left(\frac{N L}{\mu} \log(1/\epsilon)\right)$.
- Accelerated: $O\left(\frac{N \sqrt{L}}{\mu} \log(1/\epsilon)\right)$.
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 $f'_{i,t}$ evaluations to reach $\epsilon$:
  - Stochastic: $O\left(\frac{L}{\mu} (1/\epsilon)\right)$.
  - Gradient: $O(N \frac{L}{\mu} \log(1/\epsilon))$.
  - Accelerated: $O\left(N \sqrt{\frac{L}{\mu}} \log(1/\epsilon)\right)$.
  - SAG: $O\left(\max\{N, \frac{L}{\mu}\} \log(1/\epsilon)\right)$. 
Comparing Deterministic and Stochastic Methods

- quantum \((n = 50000, p = 78)\) and rcv1 \((n = 697641, p = 47236)\)
SAG Compared to FG and SG Methods

- quantum \((n = 50000, p = 78)\) and rcv1 \((n = 697641, p = 47236)\)
Other Linearly-Convergent Stochastic Methods

- Subsequent stochastic algorithms with linear rates:
  - Stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
  - Incremental surrogate optimization [Mairal, 2013].
  - **Stochastic variance-reduced gradient (SVRG)**
    - [Johnson & Zhang, 2013, Konecny & Richtarik, 2013, Mahdavi et al., 2013, Zhang et al., 2013]
  - SAGA [Defazio et al., 2014]
Other Linearly-Convergent Stochastic Methods

- Subsequent stochastic algorithms with linear rates:
  - Stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
  - Incremental surrogate optimization [Mairal, 2013].
  - **Stochastic variance-reduced gradient (SVRG)**
    [Johnson & Zhang, 2013, Konecny & Richtarik, 2013, Mahdavi et al., 2013, Zhang et al., 2013]
  - SAGA [Defazio et al., 2014]

- **SVRG has a much lower memory requirement** (later in talk).
- There are also projected/proximal/ADMM extensions.
SAG Implementation Issues

- Basic SAG algorithm:
  - while(1)
  - Sample $i$ from $\{1, 2, \ldots, N\}$.
  - Compute $f'_i(x)$.
  - $d = d - y_i + f'_i(x)$.
  - $y_i = f'_i(x)$.
  - $x = x - \frac{\alpha}{N} d$. 

- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
  - Commonly use an adaptive step-size procedure to estimate $L$.
  - Termination criterion.
    - Can use $\|x_{t+1} - x_{t}\|/\alpha = 1/n_d \approx \|\nabla f(x_t)\|$ to decide when to stop.
  - Acceleration [Lin et al., 2015].
  - Adaptive non-uniform sampling [Schmidt et al., 2013].
Basic SAG algorithm:

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

Practical variants of the basic algorithm allow:

- Regularization.
- Sparse gradients.
- Automatic step-size selection.
  - Common to use adaptive step-size procedure to estimate $L$.
- Termination criterion.
  - Can use $\|x^{t+1} - x^t\|/\alpha \approx \|\nabla f(x^t)\|$ to decide when to stop.
- Acceleration [Lin et al., 2015].
Basic SAG algorithm:

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

Practical variants of the basic algorithm allow:

- Regularization.
- Sparse gradients.
- Automatic step-size selection.
  - Common to use adaptive step-size procedure to estimate $L$.
- Termination criterion.
  - Can use $\|x^{t+1} - x^t\|/\alpha = \frac{1}{n}d \approx \|\nabla f(x^t)\|$ to decide when to stop.
- Acceleration [Lin et al., 2015].
- Adaptive non-uniform sampling [Schmidt et al., 2013].
Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
Reshuffling and Non-Uniform Sampling

- Does **re-shuffling** and doing full passes work better?
  - For classic SG: **Maybe**?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

[Recht & Ré, 2012]
Reshuffling and Non-Uniform Sampling

Does re-shuffling and doing full passes work better?

- For classic SG: Maybe?
  - Noncommutative arithmetic-geometric mean inequality conjecture.
  - [Recht & Ré, 2012]

- For SAG: NO.
  - Performance is intermediate between IAG and SAG.
Reshuffling and Non-Uniform Sampling

- Does **re-shuffling** and doing full passes work better?
  - For classic SG: **Maybe**?
    - Noncommutative arithmetic-geometric mean inequality conjecture.
  - For SAG: **NO**.
    - Performance is intermediate between IAG and SAG.
- Can **non-uniform** sampling help?

[Recht & Ré, 2012]
Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: *Maybe?*
    - Noncommutative arithmetic-geometric mean inequality conjecture. [Recht & Ré, 2012]
  - For SAG: *NO.*
    - Performance is intermediate between IAG and SAG.

- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.
Reshuffling and Non-Uniform Sampling

- Does **reshuffling** and doing full passes work better?
  - For classic SG: **Maybe?**
    - Noncommutative arithmetic-geometric mean inequality conjecture. [Recht & Ré, 2012]
  - For SAG: **NO.**
    - Performance is intermediate between IAG and SAG.

- Can **non-uniform** sampling help?
  - For classic SG methods, can only improve constants.
  - For SAG, **bias sampling towards Lipschitz constants** $L_i$,
    \[
    \|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|.
    \]

improves rate to depend on $L_{\text{mean}}$ instead of $L_{\text{max}}$.

(with bigger step size)
Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture. [Recht & Ré, 2012]
  - For SAG: NO.
    - Performance is intermediate between IAG and SAG.

- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.
  - For SAG, bias sampling towards Lipschitz constants $L_i$,
    \[
    \|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|.
    \]
    improves rate to depend on $L_{\text{mean}}$ instead of $L_{\text{max}}$. (with bigger step size)
  - Adaptively estimate $L_i$ as you go. (see paper/code).
  - Slowly learns to ignore well-classified examples.
SAG with Adaptive 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 \((n = 12678, p = 4932)\)

- Adaptive non-uniform sampling helps a lot.
SAG with Mini-Batches

- Reasons to use mini-batches with SAG:
  1. Parallelize gradient calculation.
  2. Decrease memory (only store gradient of the mini-batch).

Convergence rate depends on $L$ for mini-batches:

$$L(B) \leq L(i),$$

possibly by up to $|B|$. Allows bigger step-size, $\alpha = \frac{1}{L(B)}$. Place examples in batches to make $L(B)$ small.
SAG with Mini-Batches

Reasons to use mini-batches with SAG:

1. Parallelize gradient calculation.
2. Decrease memory (only store gradient of the mini-batch).
3. Increase convergence rate.
   (classic SG methods: only changes constant)
SAG with Mini-Batches

- Reasons to use mini-batches with SAG:
  1. Parallelize gradient calculation.
  2. Decrease memory (only store gradient of the mini-batch).
  3. Increase convergence rate.
     (classic SG methods: only changes constant)

- Convergence rate depends on $L$ for mini-batches:
  - $L(B) \leq L(i)$, possibly by up to $|B|$.
  - Allows bigger step-size, $\alpha = 1/L(B)$.
  - Place examples in batches to make $L(B)$ small.
Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
- Besides mini-batches, structure in objective may avoid this:
  - For linear models where \( f_i(w) = g(w^T x_i) \), then only require \( O(n) \) memory:
    \[
    \nabla f_i(w) = g'(w^T x_i) x_i.
    \]
Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the **memory requirement**.
- Besides mini-batches, structure in objective may avoid this:
  - For linear models where $f_i(w) = g(w^T x_i)$, then only require $O(n)$ memory:
    $$\nabla f_i(w) = g'(w^T x_i) x_i.\$$
  - For CRFs, only need to store marginals of parts.

(optical character and named-entity recognition tasks)
A major disadvantage of SAG is the memory requirement. Besides mini-batches, structure in objective may avoid this:

- For linear models where $f_i(w) = g(w^T x_i)$, then only require $O(n)$ memory:
  \[
  \nabla f_i(w) = g'(w^T x_i) x_i.
  \]

- For CRFs, only need to store marginals of parts.

(optical character and named-entity recognition tasks)

If the above don’t work, use SVRG...
Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with $x_0$
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f'_i(x_s)$
  - $x^0 = x_s$
  - $x_{s+1} = x_t$ for random $t \in \{1, 2, \ldots, m\}$

Requires 2 gradients per iteration and occasional full passes, but only requires storing $d_s$ and $x_s$.
Practical issues similar to SAG (acceleration versions, automatic step-size/termination, handles sparsity/regularization, non-uniform sampling, mini-batches).
Stochastic Variance-Reduced Gradient

**SVRG algorithm:**

- 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 $t \in \{1, 2, \ldots, m\}$. 

Requires 2 gradients per iteration and occasional full passes, but only requires storing $d_s$ and $x_s$. Practical issues similar to SAG (acceleration versions, automatic step-size/termination, handles sparsity/regularization, non-uniform sampling, mini-batches).
Stochastic Variance-Reduced Gradient

**SVRG algorithm:**

- 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 $i_t \in \{1, 2, \ldots, N\}$
    - $x^t = x^{t-1} - \alpha_t (f'_{it}(x^{t-1}) - f'_{it}(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.

Requires 2 gradients per iteration and occasional full passes, but only requires storing $d_s$ and $x_s$. 
Stochastic Subgradient Convergence Rate of SSG
Practical Subgradient Methods
Stochastic Average Gradient
Infinite Data Sets

Stochastic Variance-Reduced Gradient

**SVRG algorithm:**

- 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 $t \in \{1, 2, \ldots, m\}$

Requires **2 gradients per iteration and occasional full passes**, but **only requires storing** $d_s$ and $x_s$.

Practical issues similar to SAG (acceleration versions, automatic step-size/termination, handles sparsity/regularization, non-uniform sampling, mini-batches).
Stochastic Subgradient for Infinite Datasets?

- In analysis of stochastic subgradient, two assumptions on $g_{i_t}$:
  - Unbiased approximation of subgradient: $\mathbb{E}[g_{i_t}] = g_t$.
  - Variance is bounded: $\mathbb{E}[\|g_{i_t}\|^2] \leq B^2$.

- Unlike SAG, stochastic subgradient applies to general stochastic optimization:
  $$\arg\min_{x \in \mathbb{R}^d} \mathbb{E}[f_i(x)].$$
Stochastic Subgradient for Infinite Datasets?

- In analysis of stochastic subgradient, two assumptions on $g_{it}$:
  - Unbiased approximation of subgradient: $\mathbb{E}[g_{it}] = g_t$.
  - Variance is bounded: $\mathbb{E}[\|g_{it}\|^2] \leq B^2$.

- Unlike SAG, stochastic subgradient applies to general stochastic optimization:
  $$\arg\min_{x \in \mathbb{R}^d} \mathbb{E}[f_i(x)].$$

- We can use stochastic subgradient on IID samples from infinite dataset:
  - $O(1/\epsilon)$ rate still applies.
Stochastic vs. Deterministic for Stochastic Objectives

Consider smooth/strongly-convex stochastic objectives,

\[
\min_{x \in \mathbb{R}^D} \mathbb{E}[f_i(x)],
\]

including the generalization error in machine learning.
Stochastic vs. Deterministic for Stochastic Objectives

- Consider smooth/strongly-convex stochastic objectives,
  \[
  \min_{x \in \mathbb{R}^D} \mathbb{E}[f_i(x)],
  \]
  including the generalization error in machine learning.

- Error \( \epsilon \) has two parts [Bottou & Bousquet, 2007]:
  \[
  \epsilon = (\text{optimization error}) + (\text{estimation error}).
  \]
  (for generalization error, also have model error)
Stochastic vs. Deterministic for Stochastic Objectives

- Consider smooth/strongly-convex stochastic objectives,

\[ \min_{x \in \mathbb{R}^D} \mathbb{E}[f_i(x)], \]

including the generalization error in machine learning.

- Error \( \epsilon \) has two parts [Bottou & Bousquet, 2007]:

\[ \epsilon = (\text{optimization error}) + (\text{estimation error}). \]

(for generalization error, also have model error)

- Consider two strategies:
  - Generate \( t \) samples, then minimize exactly (ERM):
    - Optimization error = 0.
    - Estimation error = \( \tilde{O}(1/t) \).
Stochastic vs. Deterministic for Stochastic Objectives

- Consider smooth/strongly-convex stochastic objectives,
  \[
  \min_{x \in \mathbb{R}^D} \mathbb{E}[f_i(x)],
  \]
  including the generalization error in machine learning.

- Error $\epsilon$ has two parts [Bottou & Bousquet, 2007]:
  \[
  \epsilon = \text{(optimization error)} + \text{(estimation error)}.
  \]
  (for generalization error, also have model error)

- Consider two strategies:
  - Generate $t$ samples, then minimize exactly (ERM):
    - Optimization error $= 0$.
    - Estimation error $= \tilde{O}(1/t)$.
  - Or just applying stochastic gradient as we go:
    - Optimization error $= O(1/t)$.
    - Estimation error $= \tilde{O}(1/t)$.
Stochastic vs. Deterministic for Stochastic Objectives

- So just go through your data once with stochastic gradient?
Stochastic vs. Deterministic for Stochastic Objectives

- So just go through your data once with stochastic gradient?
- “overwhelming empirical evidence shows that for almost all actual data, the ERM is better. However, we have no understanding of why this happens”

[Srebro & Sridharan, 2011]
Stochastic vs. Deterministic for Stochastic Objectives

- So just go through your data once with stochastic gradient?
- “overwhelming empirical evidence shows that for almost all actual data, the ERM is better. However, we have no understanding of why this happens” [Srebro & Sridharan, 2011]

- **Constants matter** in learning:
  - SG optimal in terms of sample size.
  - But not other quantities: $L, \mu, x^0$.
  - We care about multiplying test error by 2!
Stochastic vs. Deterministic for Stochastic Objectives

- So just go through your data once with stochastic gradient?
- “overwhelming empirical evidence shows that for almost all actual data, the ERM is better. However, we have no understanding of why this happens”
  
  [Srebro & Sridharan, 2011]

- **Constants matter** in learning:
  - SG optimal in terms of sample size.
  - But not other quantities: \( L, \mu, x^0 \).
  - We care about multiplying test error by 2!

- Growing-batch deterministic methods [Byrd et al., 2011].

- Or take \( t \) iterations of SAG on fixed \( N < t \) samples.
  - Optimization accuracy decreases to \( O(\rho^t) \).
  - Estimation accuracy increases to \( \tilde{O}(1/N) \).
Stochastic vs. Deterministic for Stochastic Objectives

- So just go through your data once with stochastic gradient?

- “overwhelming empirical evidence shows that for almost all actual data, the ERM is better. However, we have no understanding of why this happens”
  
  [Srebro & Sridharan, 2011]

- **Constants matter** in learning:
  - SG optimal in terms of sample size.
  - But not other quantities: $L, \mu, x^0$.
  - We care about multiplying test error by 2!

- Growing-batch deterministic methods [Byrd et al., 2011].

- Or take $t$ iterations of SAG on fixed $N < t$ samples.
  - Optimization accuracy decreases to $O(\rho^t)$.
  - Estimation accuracy increases to $\tilde{O}(1/N)$.

- SAG obtains better bounds for difficult optimization problems.
Streaming SVRG

Streaming SVRG algorithm [Frostig et al., 2015]:

- Start with $x_0$ and initial sample size $N$
Streaming SVRG

Streaming SVRG algorithm [Frostig et al., 2015]:

- Start with \( x_0 \) and initial sample size \( N \)
- for \( s = 0, 1, 2 \ldots \)
  - \( d_s = \frac{1}{N} \sum_{i=1}^{N} f_i'(x_s) \) for \( N \) fresh samples.
  - \( x^0 = x_s \)
Streaming SVRG

Streaming SVRG algorithm [Frostig et al., 2015]:

- Start with $x_0$ and initial sample size $N$
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f_i'(x_s)$ for $N$ fresh samples.
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick 1 fresh sample.
    - $x^t = x^{t-1} - \alpha_t (f_i'(x^{t-1}) - f_i'(x_s)) + d_s$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.
- Increase samples size $N$. 

Streaming SVRG is optimal in non-asymptotic regime.
Same variance as ERM (only true for avg(SG) asymptotically).
Second-order methods are not necessary.
Streaming SVRG

Streaming SVRG algorithm [Frostig et al., 2015]:

- Start with $x_0$ and initial sample size $N$
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f'_i(x_s)$ for $N$ fresh samples.
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick 1 fresh sample.
    - $x^t = x^{t-1} - \alpha_t(f'_{it}(x^{t-1}) - f'_{it}(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.
  - Increase samples size $N$.

- Streaming SVRG is optimal in non-asymptotic regime.
- Same variance as ERM (only true for avg(SG) asymptotically).
- Second-order methods are not necessary.
Constant-Step SG under Strong Assumptions

- We can beat $O(1/t)$ under stronger assumptions.
Constant-Step SG under Strong Assumptions

- We can beat $O(1/t)$ under stronger assumptions.
- E.g., Schmidt & Le Roux [2013],
  \[ \|f'_i(x)\| \leq B\|f'(x)\|. \]
- Crazy assumption: assumes $x^*$ minimizes $f_i$. 
Constant-Step SG under Strong Assumptions

- We can beat $O(1/t)$ under stronger assumptions.
- E.g., Schmidt & Le Roux [2013],
  \[ \| f'_i(x) \| \leq B \| f'(x) \|. \]
- Crazy assumption: assumes $x^*$ minimizes $f_i$.
- With $\alpha_t = \frac{1}{LB^2}$, stochastic gradient has
  \[ \mathbb{E}[f(x^t)] - f(x^*) \leq \left( 1 - \frac{\mu}{LB^2} \right)^t [f(x^0) - f(x^*)]. \]
- If you expect to over-fit, maybe constant $\alpha_t$ is enough?
Online Convex Optimization

- What if data is not IID?
Online Convex Optimization

- What if data is not IID?
- Addressed by online convex optimization (OCO) framework:
  - At time $t$, make a prediction $x^t$. [Zinkevich, 2003]
Online Convex Optimization

- What if data is not IID?
- Addressed by online convex optimization (OCO) framework:
  - At time $t$, make a prediction $x^t$.
  - Receive arbitrary convex loss $f_t$.

OCO analyzes regret,

$$\sum_{k=1}^{t} f_t(x^t) - f_t(x^*),$$

comparing vs. best fixed $x^*$ for any sequence $\{f_t\}$. 

[Zinkevich, 2003]
Online Convex Optimization

- What if data is not IID?
- Addressed by **online convex optimization** (OCO) framework: [Zinkevich, 2003]
  - At time $t$, make a prediction $x^t$.
  - Receive arbitrary convex loss $f_t$.
- OCO analyzes **regret**, 
  $$\sum_{k=1}^{t} f_t(x^t) - f_t(x^*),$$
  comparing vs. best fixed $x^*$ for any sequence $\{f_t\}$.
- SG-style methods achieve **optimal** $O(\sqrt{t})$ regret.
- Strongly-convex losses: $O(\log(t))$ regret [Hazan et al., 2006].
Online Convex Optimization

- What if data is not IID?
- Addressed by online convex optimization (OCO) framework:
  - At time $t$, make a prediction $x^t$.
  - Receive arbitrary convex loss $f_t$.
- OCO analyzes regret,

$$\sum_{k=1}^{t} f_t(x^t) - f_t(x^*),$$

comparing vs. best fixed $x^*$ for any sequence $\{f_t\}$.
- SG-style methods achieve optimal $O(\sqrt{t})$ regret.
- Strongly-convex losses: $O(\log(t))$ regret [Hazan et al., 2006].
- Variants exist see features first [Cesa-Bianchi et al., 1993].
- Bandit setting: no gradients.
Summary

Subgradients: generalize gradients for non-smooth convex functions.

Subgradient method: optimal but very slow general non-smooth method.

Stochastic subgradient method: same rate but $n$ times cheaper.

Constant step-size: subgradient quickly converges to approximate solution.

Decreasing step-size: subgradient slowly converges to exact solution.

Practical stochastic subgradient methods: Tricks like $\beta_t v_t$ allow training on huge sparse datasets.

Different step-size strategies and averaging significantly improve performance.

Stochastic average gradient: $O(\log(1/\epsilon))$ iterations with 1 gradient per iteration.

Infinite Training Sets can be used with stochastic subgradient.

But recent results indicate it's sometimes better to apply SAG to finite sample.

Next time: how to use (some) infinite sets of features.
Summary

- **Subgradients**: generalize gradients for non-smooth convex functions.
- **Subgradient method**: optimal but very-slow general non-smooth method.
- **Stochastic subgradient method**: same rate but \( n \) times cheaper.
Summary

- **Subgradients**: generalize gradients for non-smooth convex functions.
- **Subgradient method**: optimal but very-slow general non-smooth method.
- **Stochastic subgradient method**: same rate but $n$ times cheaper.
- **Constant step-size**: subgradient quickly converges to approximate solution.
- **Decreasing step-size**: subgradient slowly converges to exact solution.
- **Practical stochastic subgradient** methods:
  - Tricks like $\beta^t v^t$ allow training on huge sparse datasets.
  - Different step-size strategies and averaging significantly improve performance.
Summary

- **Subgradients**: generalize gradients for non-smooth convex functions.
- **Subgradient method**: optimal but very-slow general non-smooth method.
- **Stochastic subgradient method**: same rate but \( n \) times cheaper.
- **Constant step-size**: subgradient quickly converges to approximate solution.
- **Decreasing step-size**: subgradient slowly converges to exact solution.
- **Practical stochastic subgradient methods**:
  - Tricks like \( \beta^t v^t \) allow training on huge sparse datasets.
  - Different step-size strategies and averaging significantly improve performance.
- **Stochastic average gradient**: \( O(\log(1/\epsilon)) \) iterations with 1 gradient per iteration.
Summary

- **Subgradients**: generalize gradients for non-smooth convex functions.
- **Subgradient method**: optimal but very-slow general non-smooth method.
- **Stochastic subgradient method**: same rate but $n$ times cheaper.
- **Constant step-size**: subgradient quickly converges to approximate solution.
- **Decreasing step-size**: subgradient slowly converges to exact solution.
- **Practical stochastic subgradient methods**:
  - Tricks like $\beta^t v^t$ allow training on huge sparse datasets.
  - Different step-size strategies and averaging significantly improve performance.
- **Stochastic average gradient**: $O(\log(1/\epsilon))$ iterations with 1 gradient per iteration.
- **Infinite Training Sets** can be used with stochastic subgradient.
  - But recent results indicate it’s sometimes better to apply SAG to finite sample.
- Next time: how to use (some) infinite sets of features.