Inexact Proximal-Gradient Methods and Linearly-Convergent Stochastic-Gradient Methods

Mark Schmidt

Joint work with Nicolas Le Roux, Francis Bach, Michael Friedlander

INRIA - SIERRA Project - Team
Laboratoire d'Informatique de l'École Normale Supérieure
(CNRS/ENS/UMR 8548)

May 2012

Outline

Composite Convex Optimization Problems

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

where g and h are convex but h may be non-smooth

Composite Convex Optimization Problems

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

where g and h are convex but h may be non-smooth

- Often, g is a data-fitting term, and h is a regularizer,

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{N} l_i(x) + \lambda r(x).$$

- A well-studied example is ℓ_1-regularized least squares,

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|^2 + \lambda \|x\|_1.$$

Convergence Rates of Proximal-Gradient Methods

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

where g and h are convex but h may be non-smooth

Convergence Rates of Proximal-Gradient Methods

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

where g and h are convex but h may be non-smooth

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Stochastic Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$

Convergence Rates of Proximal-Gradient Methods

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

  where g and h are convex but h may be non-smooth

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Stochastic Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$
Proximal-Gradient	$O(1/k)$	$O((1-\gamma)^k)$

Convergence Rates of Proximal-Gradient Methods

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

where g and h are convex but h may be non-smooth

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Stochastic Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$
Proximal-Gradient	$O(1/k)$	$O((1-\gamma)^k)$
Accelerated Proximal-Gradient	$O(1/k^2)$	$O((1-\sqrt{\gamma})^k)$

Convergence Rates of Proximal-Gradient Methods

- We consider composite optimization problems:

$$\min_{x \in \mathbb{R}^d} f(x) := g(x) + h(x),$$

  where g and h are convex but h may be non-smooth

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Stochastic Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$
Proximal-Gradient	$O(1/k)$	$O((1-\gamma)^k)$
Accelerated Proximal-Gradient	$O(1/k^2)$	$O((1-\sqrt{\gamma})^k)$

- Proximal-gradient methods have the same convergence rates as [accelerated] gradient methods for smooth optimization.
  [Nesterov, 2007, Beck & Teboulle, 2009]

Non-Simple Regularizers and Big-N Problems

For many problems we can not use proximal-gradient iterations:

1. We can not efficiently compute the proximity operator.
2. We can not efficiently evaluate the gradient of g.

Non-Simple Regularizers and Big-N Problems

For many problems we can not use proximal-gradient iterations:

1. We can not efficiently compute the proximity operator.
2. We can not efficiently evaluate the gradient of g.

For example,

1. Overlapping-group ℓ_1-regularization,

$$h(x) := \lambda \sum_{g \in \mathcal{G}} \|x_g\|,$$

2. Data-fitting with a large number of samples N,

$$g(x) := \sum_{i=1}^{N} f_i(x).$$

Non-Simple Regularizers and Big-N Problems

We can often efficiently approximate these quantities:

1. For overlapping-group ℓ_1-regularization, we can use an inexact proximity operator,
$$y \approx \mathrm{prox}[x].$$

2. For data-fitting with a large number of samples N, we can use a subsample of the f_i,
$$\frac{1}{|\mathcal{B}|}\sum_{i \in \mathcal{B}} f_i'(x) \approx \frac{1}{N}\sum_{i=1}^{N} f_i'(x) = f'(x).$$

Non-Simple Regularizers and Big-N Problems

We can often efficiently approximate these quantities:

1. For overlapping-group ℓ_1-regularization, we can use an inexact proximity operator,
$$y \approx \mathrm{prox}[x].$$
2. For data-fitting with a large number of samples N, we can use a subsample of the f_i,
$$\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} f_i'(x) \approx \frac{1}{N}\sum_{i=1}^{N} f_i'(x) = f'(x).$$

But, we may lose the convergence rates with these approximations.

Overview of Contributions

This talk considers 2 cases where we can achieve fast convergence rates despite an error in the proximity or gradient calculation:

Overview of Contributions

This talk considers 2 cases where we can achieve fast convergence rates despite an error in the proximity or gradient calculation:

1. **Inexact Proximal-Gradient Methods**:
   - We show that [accelerated] proximal-gradient methods with decreasing errors achieve the rates of the error-free case.

Overview of Contributions

This talk considers 2 cases where we can achieve fast convergence rates despite an error in the proximity or gradient calculation:

1. **Inexact Proximal-Gradient Methods**:
   - We show that [accelerated] proximal-gradient methods with decreasing errors achieve the rates of the error-free case.
2. **Linearly-Convergent Stochastic-Gradient Methods**:
   - We show that using an increasing sample of the f_i functions achieves a linear convergence rate.

Overview of Contributions

This talk considers 2 cases where we can achieve fast convergence rates despite an error in the proximity or gradient calculation:

1. **Inexact Proximal-Gradient Methods**:
   - We show that [accelerated] proximal-gradient methods with decreasing errors achieve the rates of the error-free case.
2. **Linearly-Convergent Stochastic-Gradient Methods**:
   - We show that using an increasing sample of the f_i functions achieves a linear convergence rate.
   - We propose a method that achieves a linear convergence rate but only evaluates a single f_i on each iteration.

Outline

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_{x \in \mathbb{R}^d} \; g(x).$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|x - x_k\|^2.$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_{x \in \mathbb{R}^d} \; g(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \; g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \; \frac{1}{2}\|x - (x_k - \alpha_k g'(x_k))\|^2.$$

Overview of the Basic Gradient Method

- We want to solve a smooth optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k} \|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \frac{1}{2} \|x - (x_k - \alpha_k g'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha_k g'(x_k).$$

Overview of the Basic *Proximal*-Gradient Method

- We want to solve a smooth optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k} \|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \frac{1}{2} \|x - (x_k - \alpha_k g'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha_k g'(x_k).$$

Overview of the Basic *Proximal*-Gradient Method

- We want to solve a composite optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x) + h(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|x - x_k\|^2.$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \frac{1}{2}\|x - (x_k - \alpha_k g'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha_k g'(x_k).$$

Overview of the Basic *Proximal*-Gradient Method

- We want to solve a composite optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x) + h(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|x - x_k\|^2 + h(x).$$

- We can equivalently write this as the quadratic optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \frac{1}{2}\|x - (x_k - \alpha_k g'(x_k))\|^2.$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha_k g'(x_k).$$

Overview of the Basic *Proximal*-Gradient Method

- We want to solve a composite optimization problem,

$$\min_{x\in\mathbb{R}^d} g(x)+h(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x\in\mathbb{R}^d} g(x_k)+\langle g'(x_k), x-x_k\rangle + \frac{1}{2\alpha_k}\|x-x_k\|^2+h(x).$$

- We can equivalently write this as the proximal optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x\in\mathbb{R}^d} \frac{1}{2}\|x-(x_k-\alpha_k g'(x_k))\|^2+\alpha_k h(x).$$

- The solution is the gradient algorithm:

$$x_{k+1} = x_k - \alpha_k g'(x_k).$$

Overview of the Basic *Proximal*-Gradient Method

- We want to solve a composite optimization problem,

$$\min_{x \in \mathbb{R}^d} g(x) + h(x).$$

- At iteration x_k we use a *quadratic upper bound* on g,

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} g(x_k) + \langle g'(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|x - x_k\|^2 + h(x).$$

- We can equivalently write this as the proximal optimization

$$x_{k+1} = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \frac{1}{2}\|x - (x_k - \alpha_k g'(x_k))\|^2 + \alpha_k h(x).$$

- The solution is the proximal-gradient algorithm:

$$x_{k+1} = \mathrm{prox}_{\alpha_k}[x_k - \alpha_k g'(x_k)].$$

Special case of Projected-Gradient Methods

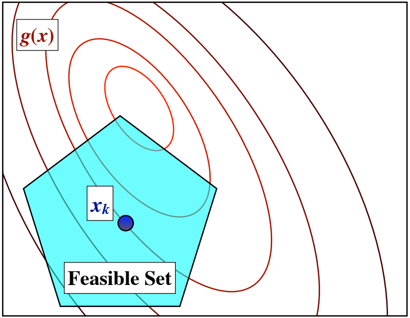
- Projected-gradient methods are a special case:

$$h(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$

Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

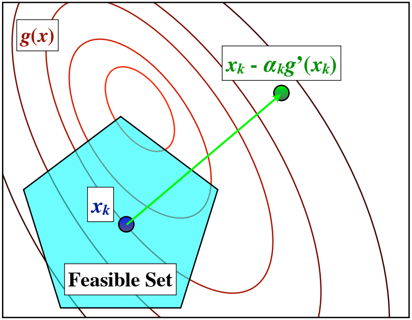
$$h(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

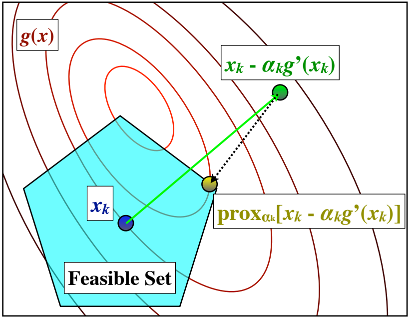
$$h(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

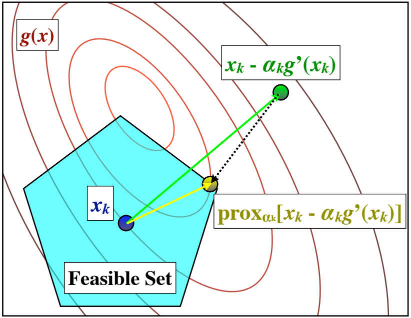
$$h(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



Special case of Projected-Gradient Methods

- Projected-gradient methods are a special case:

$$h(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$



file:///Users/Mark/Pictures/2011/11Paris/MVI_0605.MOV

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

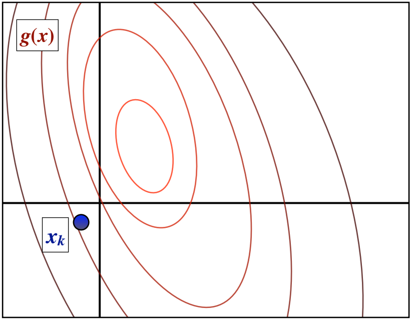
- In this case $\mathrm{prox}_{\alpha_k}[x]_i$ shrinks $|x_i|$ by $\min\{\alpha_k \lambda, |x_i|\}$

Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

- In this case $\text{prox}_{\alpha_k}[x]_i$ shrinks $|x_i|$ by $\min\{\alpha_k \lambda, |x_i|\}$

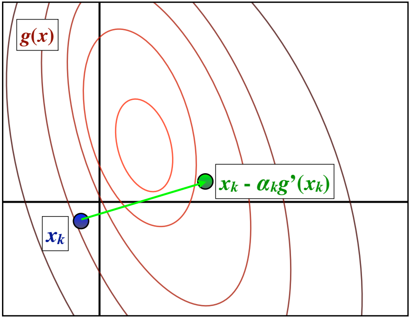


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

- In this case $\mathrm{prox}_{\alpha_k}[x]_i$ shrinks $|x_i|$ by $\min\{\alpha_k \lambda, |x_i|\}$

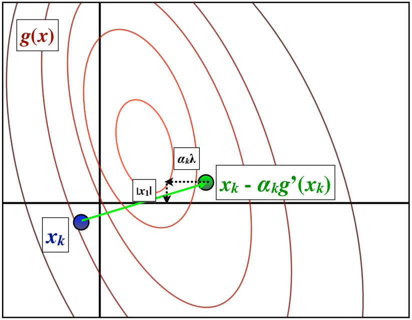


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

- In this case $\text{prox}_{\alpha_k}[x]_i$ shrinks $|x_i|$ by $\min\{\alpha_k \lambda, |x_i|\}$

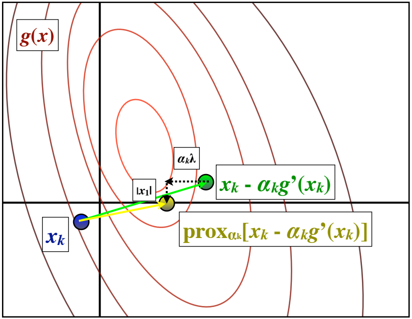


Special case of Iterative Soft-Thresholding Methods

- Iterative Soft-Thresholding methods are a special case:

$$h(x) = \lambda \|x\|_1.$$

- In this case $\mathrm{prox}_{\alpha_k}[x]_i$ shrinks $|x_i|$ by $\min\{\alpha_k \lambda, |x_i|\}$



file:///Users/Mark/Pictures/2011/12Paris/MVI_0643.MOV

Accelerated (Proximal-)Gradient Methods

- Proximal-gradient methods have the same convergence rates as gradient methods for smooth optimization.

Accelerated (Proximal-)Gradient Methods

- Proximal-gradient methods have the same convergence rates as gradient methods for smooth optimization.
- But for smooth problems accelerated gradient methods have faster rates [Nesterov, 1983]:

$$
\begin{aligned}
x_{k+1} &= y_k - \alpha_k g'(y_k), \\
y_{k+1} &= x_{k+1} + \beta_k(x_{k+1} - x_k).
\end{aligned}
$$

Accelerated (Proximal-)Gradient Methods

- Proximal-gradient methods have the same convergence rates as gradient methods for smooth optimization.
- But for smooth problems accelerated gradient methods have faster rates [Nesterov, 1983]:

$$x_{k+1} = y_k - \alpha_k g'(y_k),$$
$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

- For composite problems accelerated proximal-gradient methods have these same rates:

$$x_{k+1} = \mathrm{prox}_{\alpha_k}[y_k - \alpha_k g'(y_k)],$$
$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

Exact Proximal-Gradient Methods

- For what problems can we apply proximal-gradient methods?

Exact Proximal-Gradient Methods

- For what problems can we apply proximal-gradient methods?
- We can efficiently compute the proximity operator for:
    1. *ℓ_1-Regularization*.
    2. *Group ℓ_1-Regularization*.
    3. *Lower and upper bound constraints*.
    4. *Hyper-plane and half-space constraints*.
    5. *Simplex constraints*.
    6. *Euclidean cone constraints*.

Exact Proximal-Gradient Methods

- For what problems can we apply proximal-gradient methods?
- We can efficiently compute the proximity operator for:
    1. ℓ_1-*Regularization*.
    2. *Group* ℓ_1-*Regularization*.
    3. *Lower and upper bound constraints*.
    4. *Hyper-plane and half-space constraints*.
    5. *Simplex constraints*.
    6. *Euclidean cone constraints*.
- But for many problems we can not efficiently compute the proximity operator.

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximity operator for:
  1. *Total-variation regularization and generalizations like the graph-guided fused-LASSO*.
  2. *Nuclear-norm regularization and other regularizers on the singular values of matrices*.
  3. *Overlapping group ℓ_1-regularization with general groups*.
  4. *Positive semi-definite cone*.
  5. *Combinations of simple functions*.

Summary of Contribution

Many recent works use inexact proximal-gradient methods:

- Cai et al. [2010], Liu & Ye [2010], Schmidt & Murphy [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011].

Summary of Contribution

Many recent works use inexact proximal-gradient methods:

- Cai et al. [2010], Liu & Ye [2010], Schmidt & Murphy [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011].

Our question:

- Can inexact proximal-gradient methods achieve the fast convergence rates?

Summary of Contribution

Many recent works use inexact proximal-gradient methods:

- Cai et al. [2010], Liu & Ye [2010], Schmidt & Murphy [2010], Barbero & Sra [2011], Fadili & Peyré [2011], Ma et al. [2011].

Our question:

- Can inexact proximal-gradient methods achieve the fast convergence rates?

Our contribution:

- *Inexact proximal-gradient methods can achieve the fast convergence rates, if the errors are appropriately controlled*.

Outline

Prior Work on Proximal-Gradient Methods with Error

- Proximal-gradient methods with zero-mean random error:
  [Duchi & Singer, 2009, Langford et al., 2009]
  - Same slow convergence rates as sub-gradient methods.

Prior Work on Proximal-Gradient Methods with Error

- Proximal-gradient methods with zero-mean random error:
  [Duchi & Singer, 2009, Langford et al., 2009]
  - Same slow convergence rates as sub-gradient methods.
- Projected-gradient methods with fixed error magnitude:
  [Nedic & Bertsekas, 2000, d'Aspremont, 2008, Baes, 2009, Devolder et al., 2011]
  - Fast convergence rate up to some fixed error level.

Prior Work on Proximal-Gradient Methods with Error

- Proximal-gradient methods with zero-mean random error:
  [Duchi & Singer, 2009, Langford et al., 2009]
  - Same slow convergence rates as sub-gradient methods.
- Projected-gradient methods with fixed error magnitude:
  [Nedic & Bertsekas, 2000, d'Aspremont, 2008, Baes, 2009, Devolder et al., 2011]
  - Fast convergence rate up to some fixed error level.
- Projected-gradient methods with decreasing error magnitude:
  [Luo & Tseng, 1993, Baes, 2009, Devolder et al., 2011, Friedlander & Schmidt, 2011]
  - Either do not consider acceleration, assume an exact projection, or require that the domain is compact.

Prior Work on Proximal-Gradient Methods with Error

- Proximal-gradient methods with zero-mean random error:
  [Duchi & Singer, 2009, Langford et al., 2009]
  - Same slow convergence rates as sub-gradient methods.
- Projected-gradient methods with fixed error magnitude:
  [Nedic & Bertsekas, 2000, d'Aspremont, 2008, Baes, 2009, Devolder et al., 2011]
  - Fast convergence rate up to some fixed error level.
- Projected-gradient methods with decreasing error magnitude:
  [Luo & Tseng, 1993, Baes, 2009, Devolder et al., 2011, Friedlander & Schmidt, 2011]
  - Either do not consider acceleration, assume an exact projection, or require that the domain is compact.
- Proximal-gradient methods with decreasing error magnitude:
  [Patriksson, 1995, Combettes, 2004]
  - Do not consider convergence rates.

Problem Setting and Algorithm

- We consider the problem

$$\min_{x \in \mathbb{R}^d} \; g(x) + h(x).$$

- The basic proximal-gradient method uses

$$x_k = \mathrm{prox}_{\alpha_k}[x_{k-1} - \alpha_k g'(x_{k-1})].$$

Problem Setting and Algorithm

- We consider the problem

$$\min_{x \in \mathbb{R}^d} \; g(x) + h(x).$$

- The basic proximal-gradient method uses

$$x_k = \operatorname{prox}_{\alpha_k}[x_{k-1} - \alpha_k g'(x_{k-1})].$$

- The accelerated proximal-gradient method uses

$$x_k = \operatorname{prox}_{\alpha_k}[y_{k-1} - \alpha_k g'(y_{k-1})],$$

where

$$y_k = x_k + \beta_k(x_k - x_{k-1}),$$

and the sequence $\{\beta_k\}$ is chosen to give a faster rate.

Central Assumptions and Notation

- In all our results we assume:
  - g is convex and g' is L-Lipschitz continuous,

$$||g'(x) - g'(y)|| \leq L||x - y||, \forall x, y.$$

(if *twice-differentiable*, equivalent to $0 \preceq g''(x) \preceq LI, \forall x$)

Central Assumptions and Notation

- In all our results we assume:
  - g is convex and g' is L-Lipschitz continuous,

$$||g'(x) - g'(y)|| \leq L||x - y||, \forall x, y.$$

    (if *twice-differentiable*, equivalent to $0 \preceq g''(x) \preceq LI, \forall x$)
  - h is a lower semi-continuous proper convex function (includes all real-valued functions, and indicator functions).

Central Assumptions and Notation

- In all our results we assume:
  - g is convex and g' is L-Lipschitz continuous,

$$||g'(x) - g'(y)|| \leq L||x - y||, \forall x, y.$$

    (if *twice-differentiable*, equivalent to $0 \preceq g''(x) \preceq LI, \forall x$)
  - h is a lower semi-continuous proper convex function (includes all real-valued functions, and indicator functions).
  - $g + h$ attains its minimum at a certain x_*.
  - The step size α_k is set to $1/L$.

Central Assumptions and Notation

- In all our results we assume:
  - g is convex and g' is L-Lipschitz continuous,

$$||g'(x) - g'(y)|| \leq L||x - y||, \forall x, y.$$

    (if *twice-differentiable*, equivalent to $0 \preceq g''(x) \preceq LI, \forall x$)
  - h is a lower semi-continuous proper convex function (includes all real-valued functions, and indicator functions).
  - $g + h$ attains its minimum at a certain x_*.
  - The step size α_k is set to $1/L$.
  - The gradient g' is computed with an error e_k.
  - x_k is an ε_k-approximate solution of the proximity operator,

$$\frac{L}{2}||x_k - y||^2 + h(x_k) \leq \varepsilon_k + \min_{x \in \mathbb{R}^d} \left\{ \frac{L}{2}||x - y||^2 + h(x) \right\}.$$

    (we can use a duality gap to check this condition)

Fast Convergence Rates of Proximal-Gradient Methods

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$
Proximal-Gradient	$O(1/k)$	$O((1-\mu/L)^k)$
Accelerated Proximal-Gradient	$O(1/k^2)$	$O((1-\sqrt{\mu/L})^k)$

Fast Convergence Rates of Proximal-Gradient Methods

- Convergence rates of methods for composite optimization:

Algorithm	Convex	Strongly Convex
Sub-Gradient	$O(1/\sqrt{k})$	$O(1/k)$
Proximal-Gradient	$O(1/k)$	$O((1-\mu/L)^k)$
Accelerated Proximal-Gradient	$O(1/k^2)$	$O((1-\sqrt{\mu/L})^k)$

- We give conditions on the sequences of gradient errors $\{e_k\}$ and proximity errors $\{\varepsilon_k\}$ that preserve these rates.

Convexity - Basic Proximal-Gradient Method

Proposition 1. *If the sequences $\{||e_k||\}$ and $\{\sqrt{\varepsilon_k}\}$ are summable then the basic proximal-gradient method achieves*

$$f\left(\frac{1}{k}\sum_{i=1}^{k} x_i\right) - f(x_*) = O(1/k).$$

Convexity - Basic Proximal-Gradient Method

Proposition 1. *If the sequences $\{||e_k||\}$ and $\{\sqrt{\varepsilon_k}\}$ are summable then the basic proximal-gradient method achieves*

$$f\left(\frac{1}{k}\sum_{i=1}^{k} x_i\right) - f(x_*) = O(1/k).$$

- E.g., $\|e_k\|$ and $\sqrt{\varepsilon_k}$ could decrease as $O(1/k^{1+\delta})$ for $\delta > 0$.

Convexity - Basic Proximal-Gradient Method

Proposition 1. *If the sequences* $\{||e_k||\}$ *and* $\{\sqrt{\varepsilon_k}\}$ *are summable then the basic proximal-gradient method achieves*

$$f\left(\frac{1}{k}\sum_{i=1}^{k} x_i\right) - f(x_*) = O(1/k).$$

- E.g., $\|e_k\|$ and $\sqrt{\varepsilon_k}$ could decrease as $O(1/k^{1+\delta})$ for $\delta > 0$.
- If they decrease as $O(1/k)$, then we get $O((\log k)^2/k)$. (see the paper for the constant factors)

Convexity - Accelerated Proximal-Gradient Method

Proposition 2. *If the sequences* $\{k||e_k||\}$ *and* $\{k\sqrt{\varepsilon_k}\}$ *are summable then the accelerated proximal-gradient method achieves*

$$f(x_k) - f(x_*) = O(1/k^2),$$

with $\beta_k = (k-1)/(k+2)$.

Convexity - Accelerated Proximal-Gradient Method

Proposition 2. *If the sequences $\{k||e_k||\}$ and $\{k\sqrt{\varepsilon_k}\}$ are summable then the accelerated proximal-gradient method achieves*

$$f(x_k) - f(x_*) = O(1/k^2),$$

with $\beta_k = (k-1)/(k+2)$.

- E.g., $\|e_k\|$ and $\sqrt{\varepsilon_k}$ could decrease as $O(1/k^{2+\delta})$ for $\delta > 0$.
- As in previous work, our analysis indicates the accelerated method is more sensitive to errors.

Strongly Convex Objectives

- We also consider the case where g is strongly convex.

Strongly Convex Objectives

- We also consider the case where g is strongly convex.
- A function g is strongly convex if the function

$$g(x) - \mu||x||^2,$$

  is convex for some $\mu > 0$.
- For *twice-differentiable* functions, equivalent to $g''(x) \succeq \mu I, \forall x$.

Strongly Convex Objectives

- We also consider the case where g is strongly convex.
- A function g is strongly convex if the function

$$g(x) - \mu||x||^2,$$

  is convex for some $\mu > 0$.
- For *twice-differentiable* functions, equivalent to $g''(x) \succeq \mu I, \forall x$.
- Here, we can obtain linear convergence rates.

Strong Convexity - Basic Proximal-Gradient Method

Proposition 3. *If the sequences* $\{||e_k||\}$ *and* $\{\sqrt{\varepsilon_k}\}$ *are in* $O(\rho^k)$ *for* $\rho < (1-\mu/L)$ *then the basic proximal-gradient method achieves*

$$||x_k - x_*|| = O((1-\mu/L)^k).$$

Strong Convexity - Basic Proximal-Gradient Method

Proposition 3. *If the sequences $\{||e_k||\}$ and $\{\sqrt{\varepsilon_k}\}$ are in $O(\rho^k)$ for $\rho < (1 - \mu/L)$ then the basic proximal-gradient method achieves*

$$||x_k - x_*|| = O((1 - \mu/L)^k).$$

- If they converge with $\rho > (1 - \mu/L)$, the rate is $O(\rho^k)$.
- If they converge with $\rho = (1 - \mu/L)$, the rate is $O(k(1 - \mu/L)^k)$.

Strong Convexity - Accelerated Method

Proposition 4. *If the sequences $\{||e_k||^2\}$ and $\{\varepsilon_k\}$ are in $O(\rho^k)$ for $\rho < (1 - \sqrt{\mu/L})$ then the accelerated proximal-gradient method achieves*

$$f(x_k) - f(x_*) = O((1 - \sqrt{\mu/L})^k),$$

with $\beta_k = (1 - \sqrt{\mu/L})/(1 + \sqrt{\mu/L})$.

Strong Convexity - Accelerated Method

Proposition 4. *If the sequences* $\{||e_k||^2\}$ *and* $\{\varepsilon_k\}$ *are in* $O(\rho^k)$ *for* $\rho < (1 - \sqrt{\mu/L})$ *then the accelerated proximal-gradient method achieves*

$$f(x_k) - f(x_*) = O((1 - \sqrt{\mu/L})^k),$$

with $\beta_k = (1 - \sqrt{\mu/L})/(1 + \sqrt{\mu/L})$.

- We also obtain a bound on the iterates because

$$\frac{\mu}{2}||x_k - x_*||^2 \leq f(x_k) - f(x_*).$$

Outline

CUR-like factorization with the ℓ_2-norm

We consider the factorization of Mairal et al. [2011] to approximate a matrix W using a subset of rows and columns:

$$\boxed{\min_X \frac{1}{2}||W - WXW||_F^2 + \lambda_{\text{row}} \sum_{i=1}^{n_r} ||X^i||_p + \lambda_{\text{col}} \sum_{j=1}^{n_c} ||X_j||_p.}$$

CUR-like factorization with the ℓ_2-norm

We consider the factorization of Mairal et al. [2011] to approximate a matrix W using a subset of rows and columns:

$$\boxed{\min_X \frac{1}{2}||W - WXW||_F^2 + \lambda_{\text{row}} \sum_{i=1}^{n_r} ||X^i||_p + \lambda_{\text{col}} \sum_{j=1}^{n_c} ||X_j||_p.}$$

- For appropriate p, yields sparse rows and sparse columns.
- Previous work used $p = \infty$, since there is no known exact algorithm for $p = 2$.

CUR-like factorization with the ℓ_2-norm

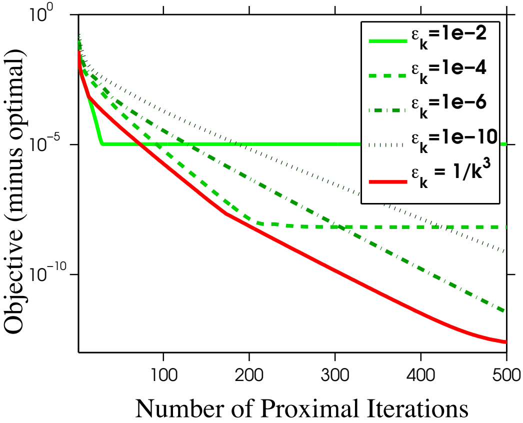
We consider the factorization of Mairal et al. [2011] to approximate a matrix W using a subset of rows and columns:

$$\boxed{\min_X \frac{1}{2}||W - WXW||_F^2 + \lambda_{\text{row}} \sum_{i=1}^{n_r} ||X^i||_p + \lambda_{\text{col}} \sum_{j=1}^{n_c} ||X_j||_p.}$$

- For appropriate p, yields sparse rows and sparse columns.
- Previous work used $p = \infty$, since there is no known exact algorithm for $p = 2$.
- We use the proximal-Dykstra algorithm to compute an approximate proximity operator with $p = 2$.
- Duality gap ensures ε_k-optimality of approximate proximity.

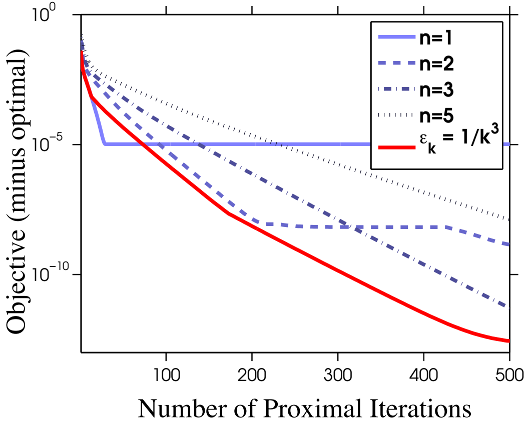
Comparison against a fixed prox solution accuracy

Using an optimal ε_k sequence compared to a fixed precision for the approximate proximity:



Comparison against a fixed number of prox iterations

Using an optimal ε_k sequence compared to running a fixed number of proximal iterations:



Discussion

- Inexact proximal-gradient methods may be useful in other applications: *total-variation or nuclear-norm regularization*.
- Our analysis also allows errors in the gradient: *undirected graphical models, kernel methods, and SDPs*.

Discussion

- Inexact proximal-gradient methods may be useful in other applications: *total-variation or nuclear-norm regularization*.
- Our analysis also allows errors in the gradient: *undirected graphical models, kernel methods, and SDPs*.
- We would like to handle an unknown L and μ.
- We would like to adaptively update $||e_k||$ and ε_k.
- We would like to analyze proximal-Newton methods.

Discussion

- Inexact proximal-gradient methods may be useful in other applications: *total-variation or nuclear-norm regularization*.
- Our analysis also allows errors in the gradient: *undirected graphical models, kernel methods, and SDPs*.
- We would like to handle an unknown L and μ.
- We would like to adaptively update $||e_k||$ and ε_k.
- We would like to analyze proximal-Newton methods.
- Villa et al. [2011] and Jiang et al. [2011] have independently analyzed accelerated proximal-gradient methods (convex g).

Summary

- Proximal-gradient methods are appealing because of their good theoretical and empirical convergence rates.

Summary

- Proximal-gradient methods are appealing because of their good theoretical and empirical convergence rates.
- But, they require the calculation of the proximity operator.

Summary

- Proximal-gradient methods are appealing because of their good theoretical and empirical convergence rates.
- But, they require the calculation of the proximity operator.
- Many authors have recently applied these methods under an inexact proximity operator.

Summary

- Proximal-gradient methods are appealing because of their good theoretical and empirical convergence rates.
- But, they require the calculation of the proximity operator.
- Many authors have recently applied these methods under an inexact proximity operator.
- We show that the convergence rates are preserved if the inexactness is appropriately controlled

Outline

Strongly Convex and Smooth Big-N Problems

- We now focus to problems of the form

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

where each f_i' is L-Lipschitz continuous and g is μ-strongly convex.

Strongly Convex and Smooth Big-N Problems

- We now focus to problems of the form

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

where each f_i' is L-Lipschitz continuous and g is μ-strongly convex.

- Includes ℓ_2-regularization of any convex loss functions,

$$f_i(x) := \frac{\lambda}{2} \|x\|^2 + l_i(x).$$

- We are interested in the case where N is large.

Stochastic Gradient Methods for Big-N Problems

- Stochastic gradient (SG) methods use iterations of the form

$$x^{k+1} = x^k - \alpha_k f'_{i_k}(x^k),$$

  where i_k is selected uniformly among the set $\{1, \dots, n\}$.
- Appealing because the iteration cost is independent of N.

Stochastic Gradient Methods for Big-N Problems

- Stochastic gradient (SG) methods use iterations of the form

$$x^{k+1} = x^k - \alpha_k f'_{i_k}(x^k),$$

  where i_k is selected uniformly among the set $\{1, \ldots, n\}$.
- Appealing because the iteration cost is independent of N.
- But SG iterations have a sublinear convergence rate

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

- This is optimal if only accessing the function through unbiased function/gradient measurements.

Full Gradient Methods for Big-N Problems

- But, for finite data sets better rates are possible.
- For example, we could use the full gradient (FG) method,

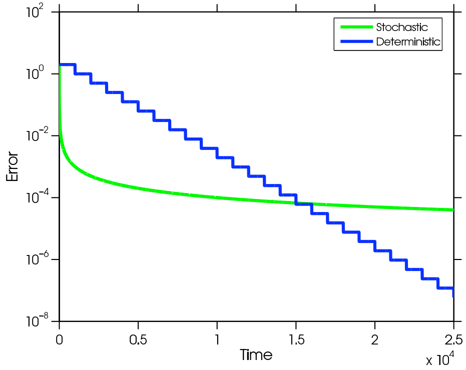
$$x^{k+1} = x^k - \alpha_k g'(x^k) = x^k - \frac{\alpha_k}{N}\sum_{i=1}^{N} f_i'(x^k).$$

- This method achieves a linear convergence rate,

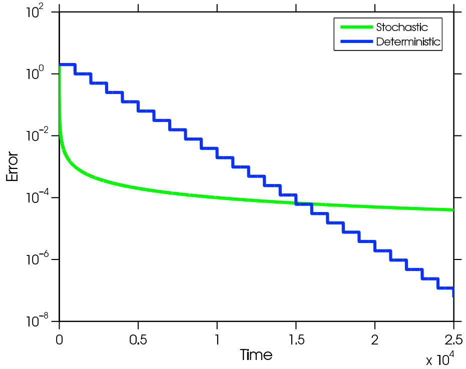
$$g(x^k) - g(x^*) = O(\rho^k).$$

- But, FG iterations are N times more expensive than SG iterations.

Stochastic vs. Full Gradient Methods

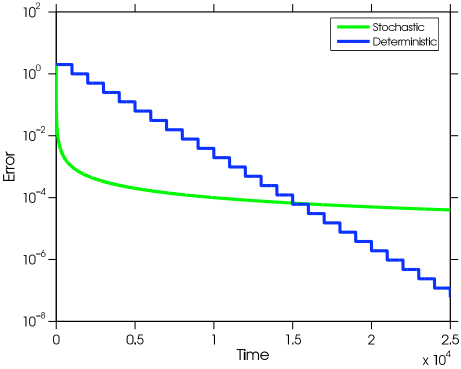


Stochastic vs. Full Gradient Methods



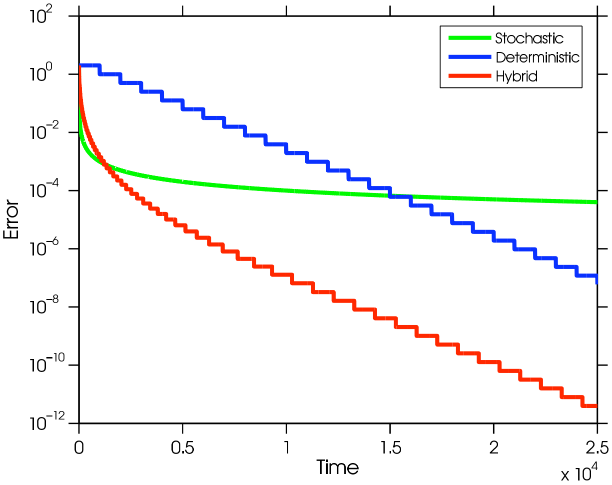
- Stochastic makes great progress initially, but slows down.
- Determinstic makes steady progress, but is expensive.

Stochastic vs. Full Gradient Methods



- Stochastic makes great progress initially, but slows down.
- Determinstic makes steady progress, but is expensive.
- Can we design hybrid methods with the best of both worlds?

Motivation for Hybrid Methods



Prior Work on Speeding up SG Methods

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

Prior Work on Speeding up SG Methods

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

- *Momentum, gradient averaging, iterate averaging, stochastic version of FG methods*:

  [Polyak & Juditsky, 1992, Tseng ,1998, Nesterov, 2009, Sunehag, 2009, Ghadimi & Lan, 2010, Xiao, 2010]

  - Can improve constants, but still have sublinear $O(1/k)$ rate.

Prior Work on Speeding up SG Methods

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

- *Momentum, gradient averaging, iterate averaging, stochastic version of FG methods*:

  [Polyak & Juditsky, 1992, Tseng ,1998, Nesterov, 2009, Sunehag, 2009, Ghadimi & Lan, 2010, Xiao, 2010]

  - Can improve constants, but still have sublinear $O(1/k)$ rate.
- *Constant step-size SG, accelerated SG*:

  [Kesten, 1958, Delyon & Juditsky, 1993, Solodov, 1998, Nedic & Bertsekas, 2000]

  - Linear convergence, but only up to a fixed tolerance.

Prior Work on Speeding up SG Methods

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

- *Momentum, gradient averaging, iterate averaging, stochastic version of FG methods*:

  [Polyak & Juditsky, 1992, Tseng ,1998, Nesterov, 2009, Sunehag, 2009, Ghadimi & Lan, 2010, Xiao, 2010]

  - Can improve constants, but still have sublinear $O(1/k)$ rate.
- *Constant step-size SG, accelerated SG*:

  [Kesten, 1958, Delyon & Juditsky, 1993, Solodov, 1998, Nedic & Bertsekas, 2000]

  - Linear convergence, but only up to a fixed tolerance.
- *Hybrid Methods, Incremental Average Gradient*:

  [Bertsekas, 1997, Blatt et al., 2008]

  - Linear rate, but iterations make full passes through the data.

Two ideas for achieving a linear rate

Is a linear rate possible, without iterations requiring full passes?

Two ideas for achieving a linear rate

Is a linear rate possible, without iterations requiring full passes?

- Idea #1: **Control the sample size** to interpolate between the stochastic and deterministic method.
  (avoids making full passes on early iterations)

Two ideas for achieving a linear rate

Is a linear rate possible, without iterations requiring full passes?

- Idea #1: **Control the sample size** to interpolate between the stochastic and deterministic method.
  (avoids making full passes on early iterations)
- Idea #2: **Build a sequence of estimates** that converge to $g'(x_k)$ as $\|x_{k-1} - x_k\| \to 0$.
  (only looks at a single f_i on each iteration)

Outline

Hybrid Deterministic-Stochastic Methods

- A common variant of SG methods uses a batch $\mathcal{B}_k$ instead of a single example,

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k} f_i'(x_k).$$

Hybrid Deterministic-Stochastic Methods

- A common variant of SG methods uses a batch $\mathcal{B}_k$ instead of a single example,

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} f_i'(x_k).$$

- The gradient error is affected by the batch size $|\mathcal{B}_k|$.
- For example, uniform sampling (without replacement) yields

$$\mathbb{E}[||e_k||^2] = \left(\frac{N - |\mathcal{B}_k|}{N}\right) \frac{S}{|\mathcal{B}_k|},$$

Hybrid Deterministic-Stochastic Methods

- A common variant of SG methods uses a batch $\mathcal{B}_k$ instead of a single example,

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} f_i'(x_k).$$

- The gradient error is affected by the batch size $|\mathcal{B}_k|$.
- For example, uniform sampling (without replacement) yields

$$\mathbb{E}[||e_k||^2] = \left(\frac{N - |\mathcal{B}_k|}{N}\right) \frac{S}{|\mathcal{B}_k|},$$

- We can choose the batch sizes to achieve linear convergence.
- Early iterations are cheap like SG iterations.

Incremental Gradient Method Error Bounds

Under standard assumptions on the f_i', by choosing $|\mathcal{B}_k|$ to satisfy

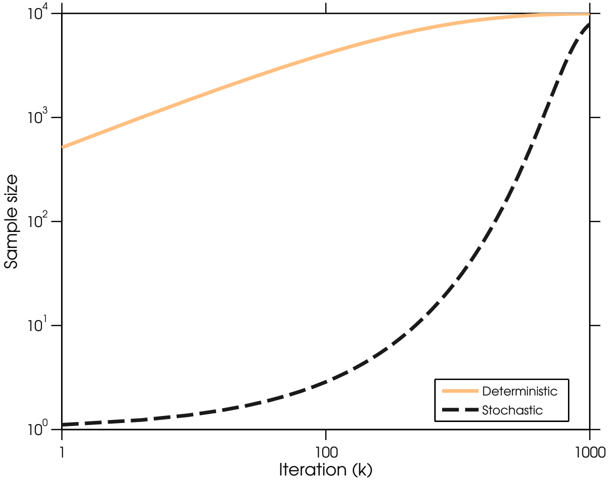
$$\frac{N - |\mathcal{B}_k|}{N} \frac{1}{|\mathcal{B}_k|} = O(\gamma^k),$$

for any $\epsilon > 0$ we have

$$\mathbb{E}[f(x_k) - f(x_*)] = [f(x_0) - f(x_*)]O([1 - \mu/L + \epsilon]^k) + O(\sigma^k),$$

where $\sigma = \max\{\gamma, 1 - \mu/L\}$.

Batch Schedule needed for Linear Rate



Improved Rates with Newton-like Scaling

- The algorithm may converge slowly if μ/L is small.
- We can also analyze a Newton-like algorithm

$$x_{k+1} = x_k + \alpha_k d_k,$$

where d_k is the solution of

$$H_k d_k = -g(x_k).$$

Improved Rates with Newton-like Scaling

- The algorithm may converge slowly if μ/L is small.
- We can also analyze a Newton-like algorithm

$$x_{k+1} = x_k + \alpha_k d_k,$$

where d_k is the solution of

$$H_k d_k = -g(x_k).$$

- We can then show rates using a modified μ and L based on the Hessian approximation H_k.

Quasi-Newton Scaling and Heuristic Line Search

- We made an implementation, where we use the *L-BFGS* quasi-Newton Hessian approximation.

Quasi-Newton Scaling and Heuristic Line Search

- We made an implementation, where we use the *L-BFGS* quasi-Newton Hessian approximation.
- To choose the step size, we use the *Armijo* condition

$$\bar{f}(x_k + \alpha_k d_k) < \bar{f}(x_k) + \eta \alpha g(x_k)^T d_k,$$

on the sampled objective

$$\bar{f}(x_k) = \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} f_i(x_k).$$

Quasi-Newton Scaling and Heuristic Line Search

- We made an implementation, where we use the *L-BFGS* quasi-Newton Hessian approximation.
- To choose the step size, we use the *Armijo* condition

$$\bar{f}(x_k + \alpha_k d_k) < \bar{f}(x_k) + \eta \alpha g(x_k)^T d_k,$$

on the sampled objective

$$\bar{f}(x_k) = \frac{1}{|\mathcal{B}_k|} \sum_{i \in \mathcal{B}_k} f_i(x_k).$$

- By increasing the batch size this eventually reduces to a conventional line-search quasi-Newton method, inheriting the global and local convergence guarantees of this method.

Batch-Size Selection in Stochastic Gradient Methods

We performed experiments comparing three algorithms:

- Deterministic: Conventional L-BFGS quasi-Newton method.
- Stochastic: Constant step-size stochastic gradient descent.
- Hybrid: An L-BFGS quasi-Newton method with batch size

$$|\mathcal{B}_{k+1}| = \lceil \min\{1.1 \cdot |\mathcal{B}_k| + 1, M\} \rceil.$$

Batch-Size Selection in Stochastic Gradient Methods

We performed experiments comparing three algorithms:

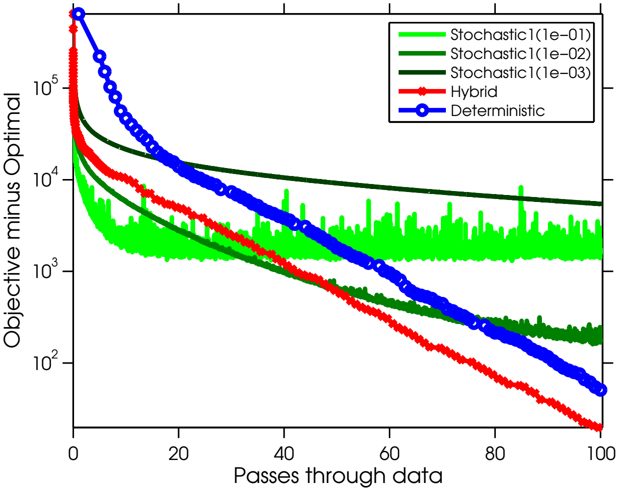
- Deterministic: Conventional L-BFGS quasi-Newton method.
- Stochastic: Constant step-size stochastic gradient descent.
- Hybrid: An L-BFGS quasi-Newton method with batch size

$$|\mathcal{B}_{k+1}| = \lceil \min\{1.1 \cdot |\mathcal{B}_k| + 1, M\} \rceil.$$

We trained a conditional random fields (CRF) on the CoNLL-2000 noun-phrase chunking shared task (chain-structure).

Evaluation on Chain-Structured CRFs

Results on chain-structured conditional random field:



Outline

Stochastic Average Gradient

- The growing-batch method eventually uses full passes on each iteration.
- Is it possible to have a linearly convergent algorithm whose iteration cost is independent of N?

Stochastic Average Gradient

- The growing-batch method eventually uses full passes on each iteration.
- Is it possible to have a linearly convergent algorithm whose iteration cost is independent of N?
    - YES!

Stochastic Average Gradient

- The growing-batch method eventually uses full passes on each iteration.
- Is it possible to have a linearly convergent algorithm whose iteration cost is independent of N?
    - YES! The stochastic average gradient (SAG) algorithm:

$$x^{k+1} = x^k - \frac{\alpha_k}{N} \sum_{i=1}^{N} y_i^k,$$

where

$$y_i^k = \begin{cases} f_i'(x_k) & \text{if } i = i_k, \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

Stochastic Average Gradient

- The growing-batch method eventually uses full passes on each iteration.
- Is it possible to have a linearly convergent algorithm whose iteration cost is independent of N?
  - YES! The stochastic average gradient (SAG) algorithm:

$$x^{k+1} = x^k - \frac{\alpha_k}{N} \sum_{i=1}^{N} y_i^k,$$

    where

$$y_i^k = \begin{cases} f_i'(x_k) & \text{if } i = i_k, \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

  - Randomized version of the incremental aggregated gradient (IAG) algorithm of Blatt et al. [2008].

Convergence Rate of SAG: Small Steps

With a step size of $\alpha_k = \frac{1}{2NL}$, the SAG iterations satisfy

$$\mathbb{E}[\|x^k - x^*\|^2] \leq C\left(1 - \frac{\mu}{8LN}\right)^k$$

- A linear rate with iterations that are independent of N!

Convergence Rate of SAG: Small Steps

With a step size of $\alpha_k = \frac{1}{2NL}$, *the SAG iterations satisfy*

$$\mathbb{E}[\|x^k - x^*\|^2] \leq C\left(1 - \frac{\mu}{8LN}\right)^k$$

- A linear rate with iterations that are independent of N!
- But, with this step size the performance is similar to the FG and IAG methods.

Convergence Rate of SAG: Big Steps

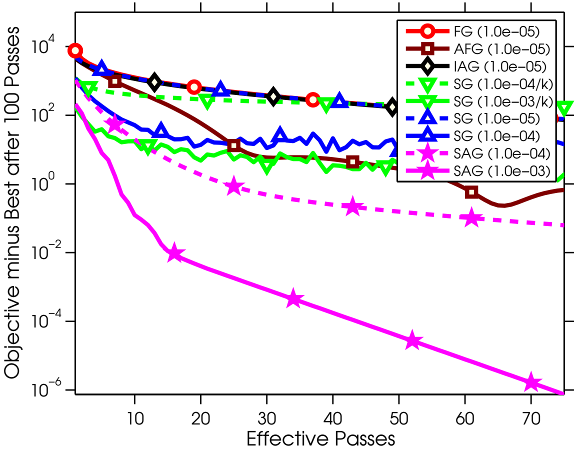
If we have enough data, the SAG iterations have a faster convergence rate with a larger step size:

If $\frac{\mu}{L} \geq \frac{8}{N}$, with a step size of $\alpha_k = \frac{1}{2N\mu}$, the SAG iterations satisfy

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

Comparison of SAG to FG and SG Methods

Comparing SAG to a variety of FG and SG methods:



Summary

Part 1:

- You can have the fast convergence rates of proximal-gradient methods, even if you can't compute the proximity operator.
    - M. Schmidt, N. Le Roux, F. Bach. **Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization**. *NIPS*, 2011.

Summary

Part 1:

- You can have the fast convergence rates of proximal-gradient methods, even if you can't compute the proximity operator.
  - M. Schmidt, N. Le Roux, F. Bach. **Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization**. *NIPS*, 2011.

Part 2:

- If you have a large finite data set, there are some options in between stochastic and exact gradient methods.
  - M. Friedlander, M. Schmidt. **Hybrid Deterministic-Stochastic Methods for Data Fitting**. *Accepted to SISC*, 2012.
  - N. Le Roux, M. Schmidt, F. Bach. **A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets**. *Submitted*, 2012.