Stochastic subgradient methods

Based on material by Mark Schmidt

Julieta Martinez

University of British Columbia

October 06, 2015

Introduction

- We are interested in a typical machine learning problem

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) + \lambda \cdot r(x)$$

data fitting term + regularizer

- Last time, we talked about gradient methods, which work when D is large
- Today we will talk about stochastic subgradient methods, which work when N is large

Introduction

- We are interested in a typical machine learning problem

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) + \lambda \cdot r(x)$$

data fitting term + regularizer

- Last time, we talked about gradient methods, which work when D is large
- Today we will talk about stochastic subgradient methods, which work when N is large

Introduction

- We are interested in a typical machine learning problem

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) + \lambda \cdot r(x)$$

data fitting term + regularizer

- Last time, we talked about gradient methods, which work when D is large
- Today we will talk about stochastic subgradient methods, which work when N is large

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$
- A deterministic gradient method computes the gradient exactly

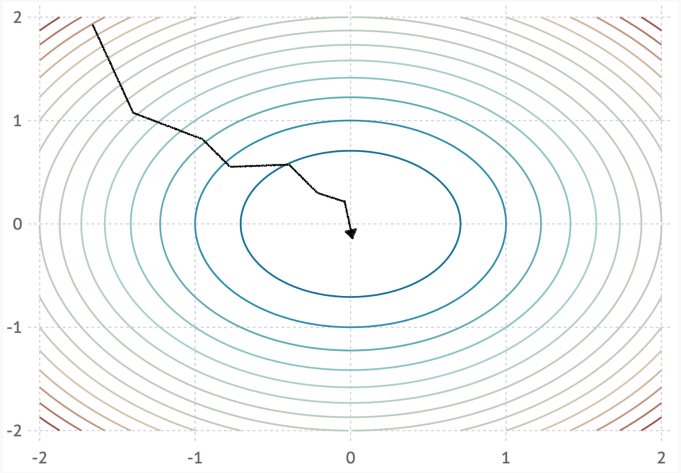
$$x_{t+1} = x_t - \alpha_t \cdot \nabla f(x_t) = x_t - \alpha_t \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i)$$

  - Computing the exact gradient is $\mathcal{O}(N)$
  - We can get convergence with constant α_t or using line-search
- A stochastic gradient method [Robbins and Monro, 1951] estimates the gradient from a sample $i_t \sim \{1, 2, \ldots, N\}$

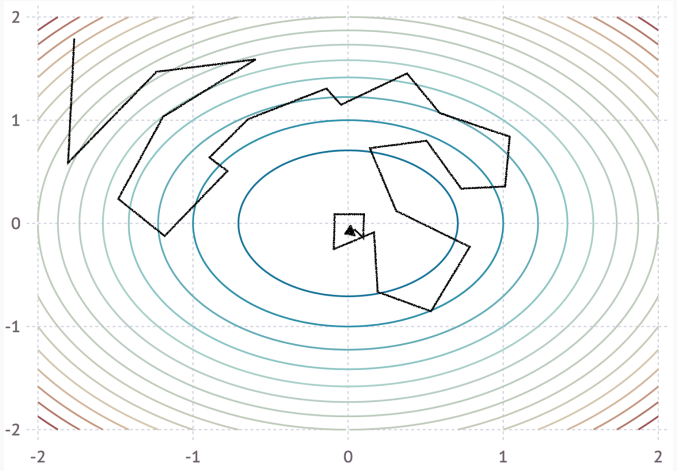
$$x_{t+1} = x_t - \alpha_t \cdot \nabla f_i(x_t) = x_t - \alpha_t \cdot \frac{1}{n}\nabla f_{i_t}(x_i)$$

  - Note that this gives an unbiased estimate of the gradient; $\mathbb{E}\left[f'_{i_t}(x)\right] = \frac{1}{N}\sum_{i=1}^{N} {}_i(x) = \nabla f(x)$.
  - The iteration cost no longer depends on N
  - Convergence requires $\alpha_t \to 0$

- We want to minimize a function $f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i)$
- Deterministic gradient methods



- Stochastic gradient methods [Robbins and Monro, 1951]



Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

Convergence

Stochastic methods are N times faster per iteration but, what about convergence?

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/t^2)$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}((1-\sqrt{u/L})^t)$	$\mathcal{O}(1/t)$

- Stochastic methods have a lower iteration cost, but a lower convergence rate
  - Sublinear rate even under strong convexity
- Bounds are unimprovable if only unbiased gradients are available
  - Momentum/acceleration does not improve convergence
  - For convergence, momentum must go to zero [Tseng, 1998]

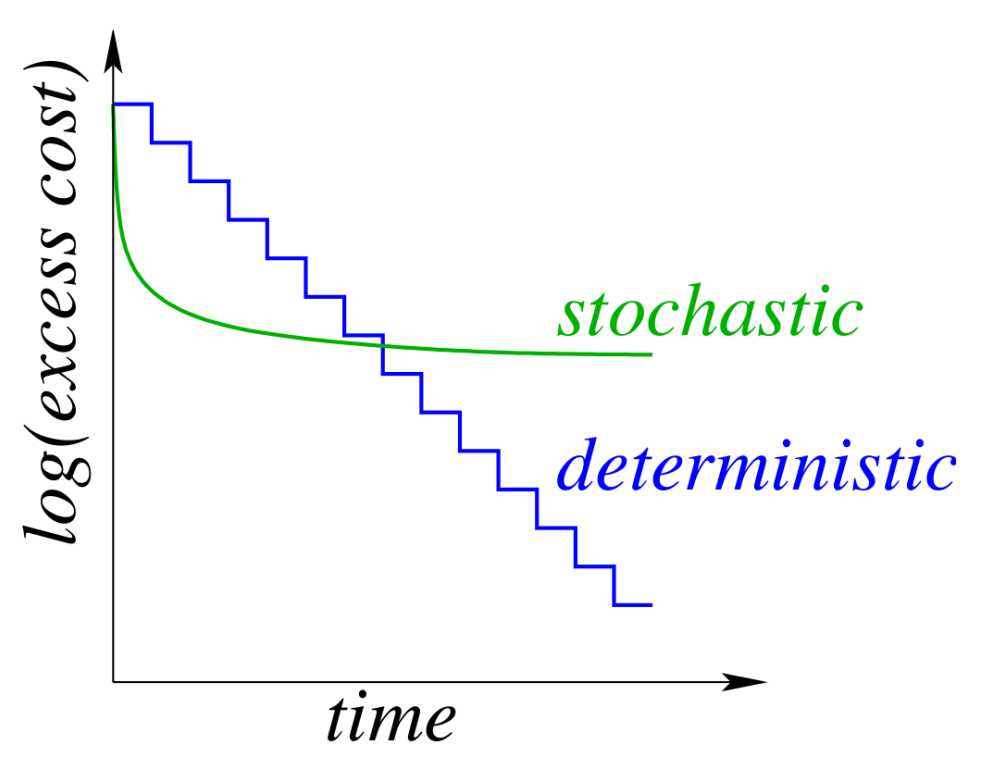


Figure : Convergence rates in the strongly convex case

- Stochastic methods are better for low-accuracy/time situations
- It can be hard to know when the crossing will happen

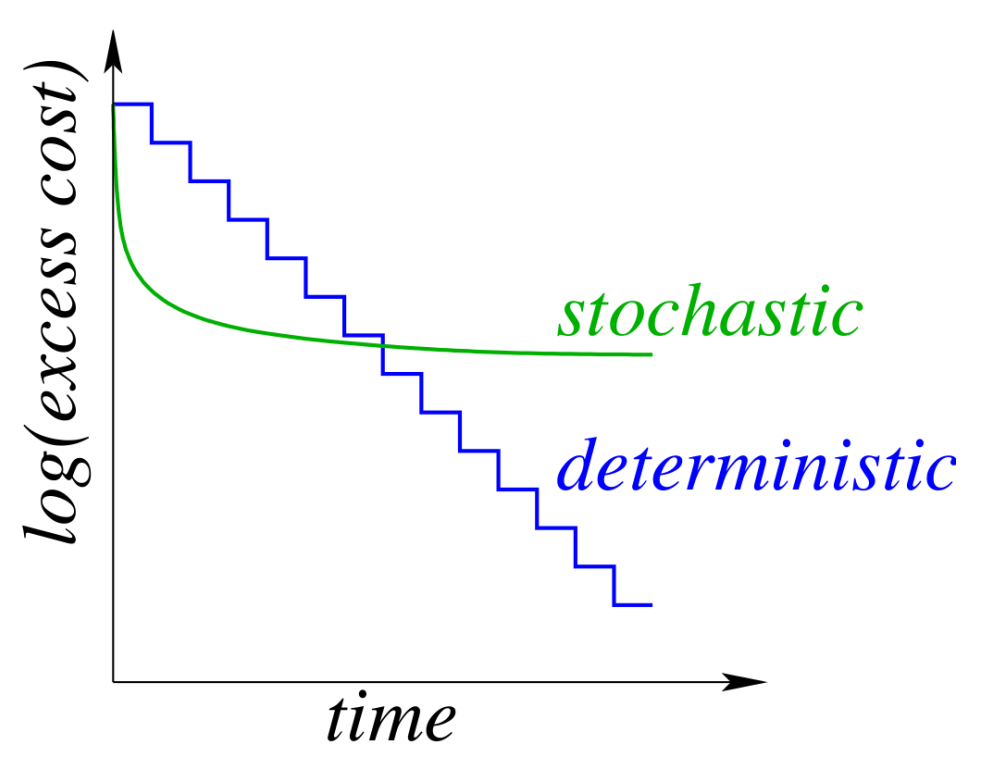


Figure : Convergence rates in the strongly convex case

- Stochastic methods are better for low-accuracy/time situations
- It can be hard to know when the crossing will happen

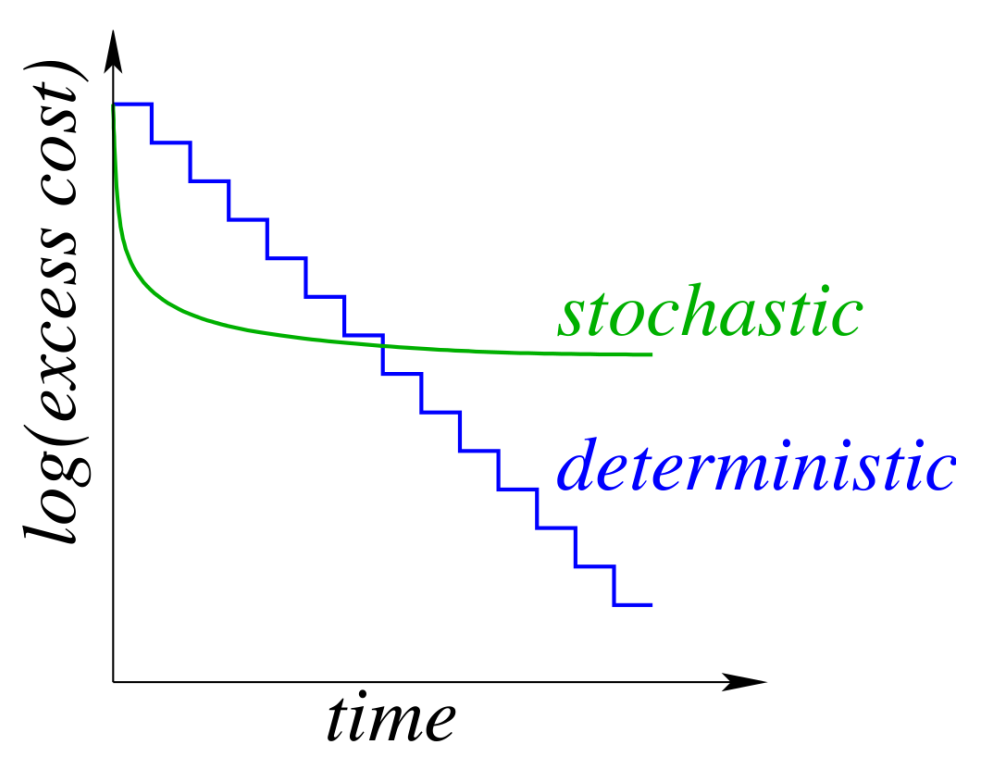


Figure : Convergence rates in the strongly convex case

- Stochastic methods are better for low-accuracy/time situations
- It can be hard to know when the crossing will happen

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^\intercal a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^T a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^T a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^T a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^\intercal a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^T a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^\intercal a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^\intercal a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- The convergence rates look quite different when the function is non-smooth
- E.g., consider the binary support vector machine

$$f(x) = \sum_{i=1}^{N} \max_x \{0, 1 - b_i(x^T a_i)\} + \lambda \|x\|^2$$

- Rates for subgradient methods in non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$\mathcal{O}(1/\sqrt{t})$	$\mathcal{O}(1/\sqrt{t})$
Strongly	$\mathcal{O}(1/t)$	$\mathcal{O}(1/t)$

- Other black-box methods such as cutting plane are not faster
- Take-away point: for non-smooth problems
  - Deterministic methods are not faster than stochastic methods
  - Stochastic methods are a free, N times faster, lunch

- For differentiable convex functions, we have

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

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

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

- At differentiable x, the only subgradient is $\nabla f(x)$
- At non-differentiable x, we have a set of subgradients, called the subdifferential, $\partial f(x)$
- Notice that if $\vec{0} \in \partial f(x)$, then x is a global minimizer

- For differentiable convex functions, we have

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

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

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

- At differentiable x, the only subgradient is $\nabla f(x)$
- At non-differentiable x, we have a set of subgradients, called the subdifferential, $\partial f(x)$
- Notice that if $\vec{0} \in \partial f(x)$, then x is a global minimizer

- For differentiable convex functions, we have

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

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

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

- At differentiable x, the only subgradient is $\nabla f(x)$
- At non-differentiable x, we have a set of subgradients, called the subdifferential, $\partial f(x)$
- Notice that if $\vec{0} \in \partial f(x)$, then x is a global minimizer

- For differentiable convex functions, we have

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

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

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

- At differentiable x, the only subgradient is $\nabla f(x)$
- At non-differentiable x, we have a set of subgradients, called the subdifferential, $\partial f(x)$
- Notice that if $\vec{0} \in \partial f(x)$, then x is a global minimizer

- For differentiable convex functions, we have

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

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

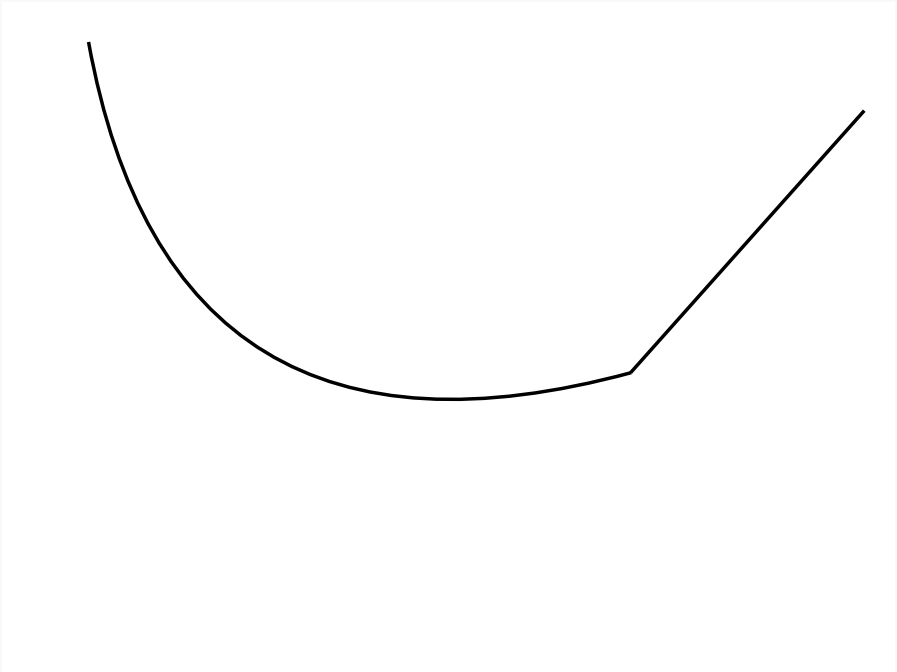
$$f(y) \geq f(x) + d^\intercal (y - x), \forall x, y.$$

- At differentiable x, the only subgradient is $\nabla f(x)$
- At non-differentiable x, we have a set of subgradients, called the subdifferential, $\partial f(x)$
- Notice that if $\vec{\mathbf{0}} \in \partial f(x)$, then x is a global minimizer

Example

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

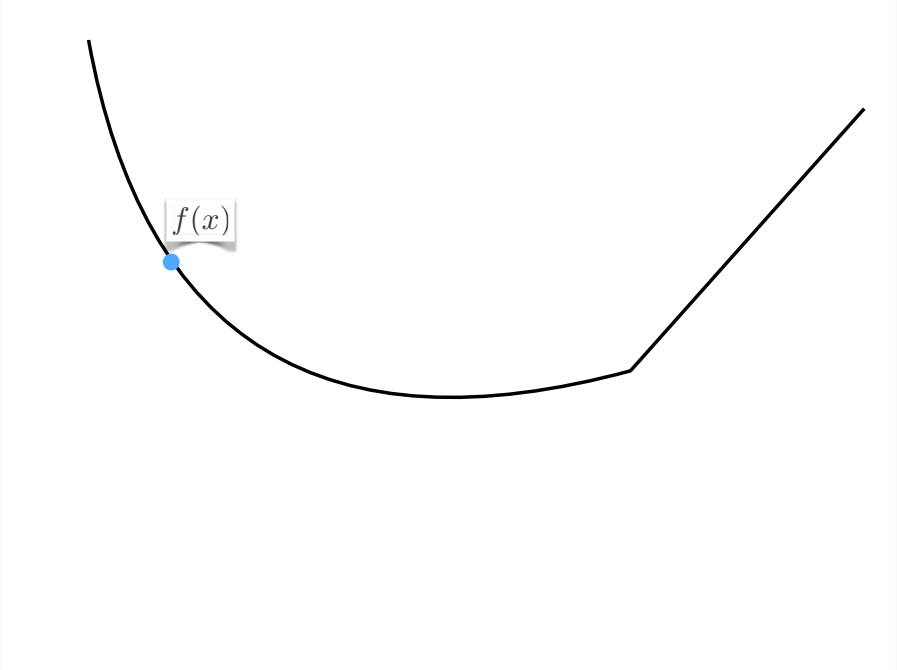
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

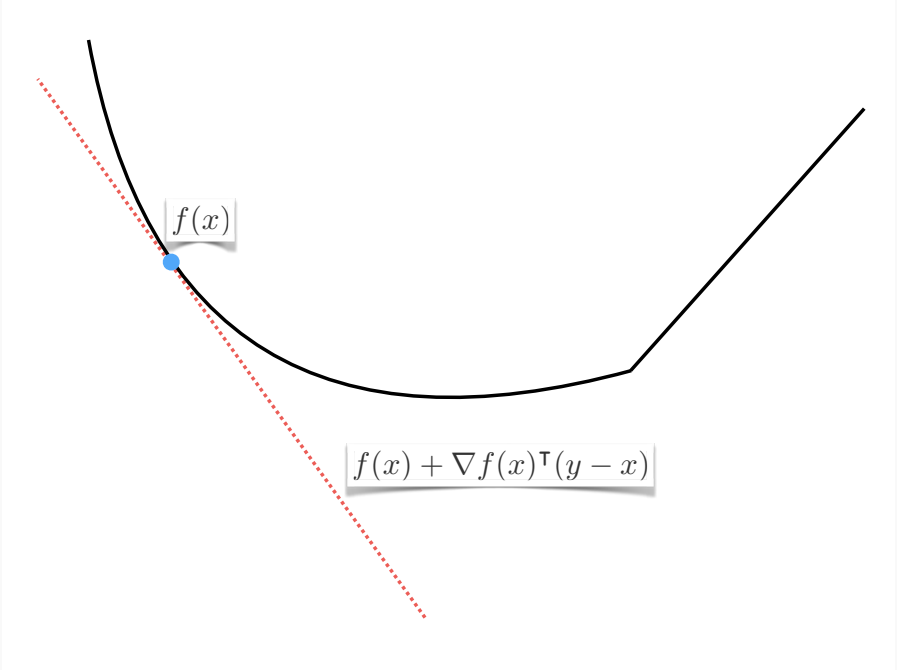
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

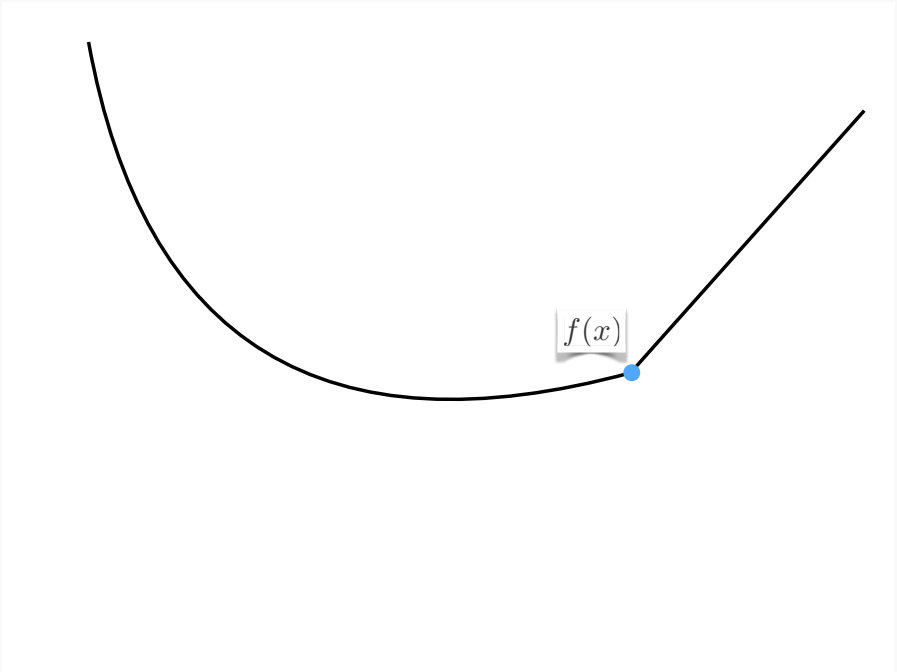
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

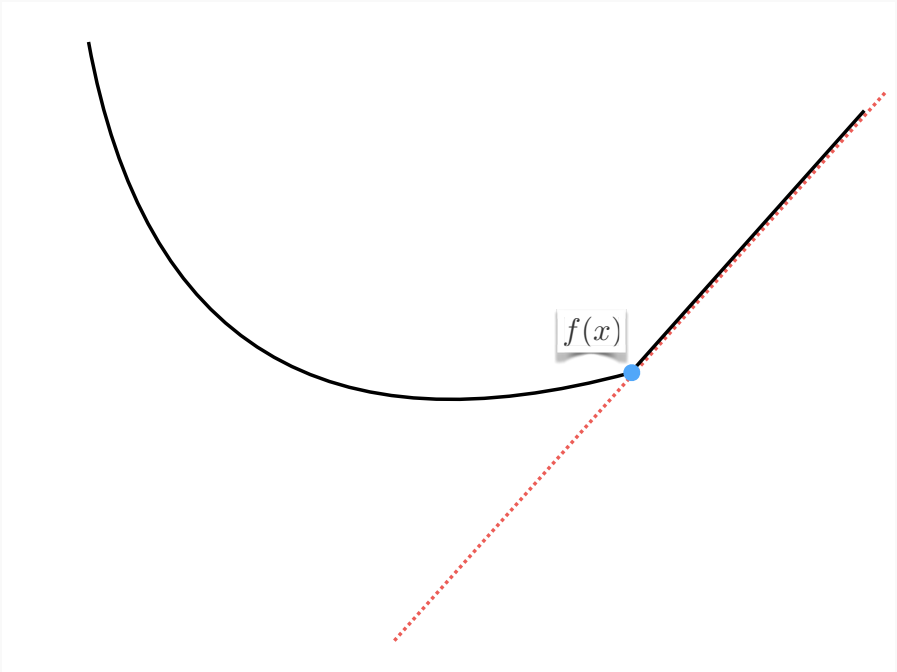
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

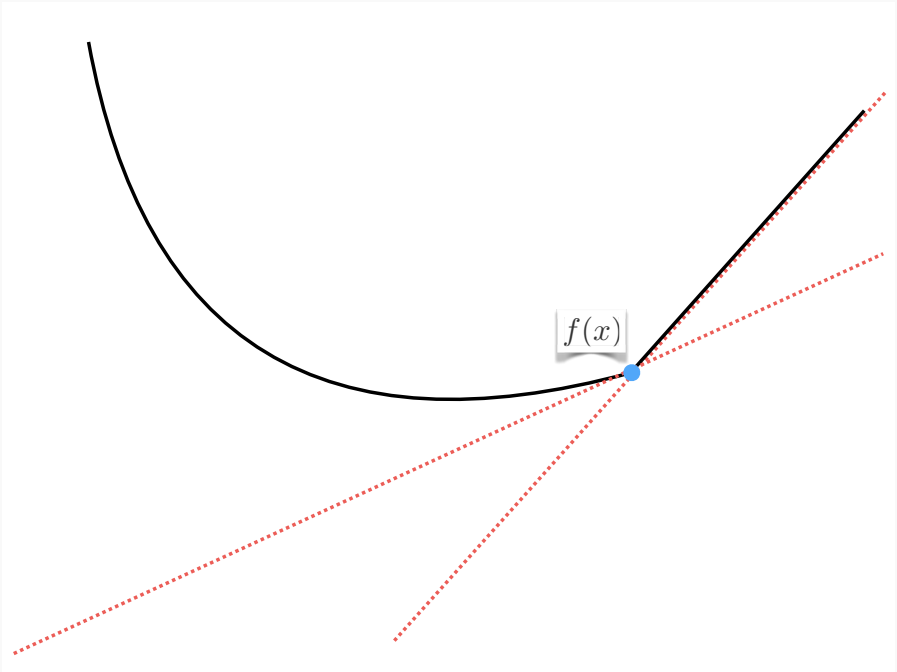
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

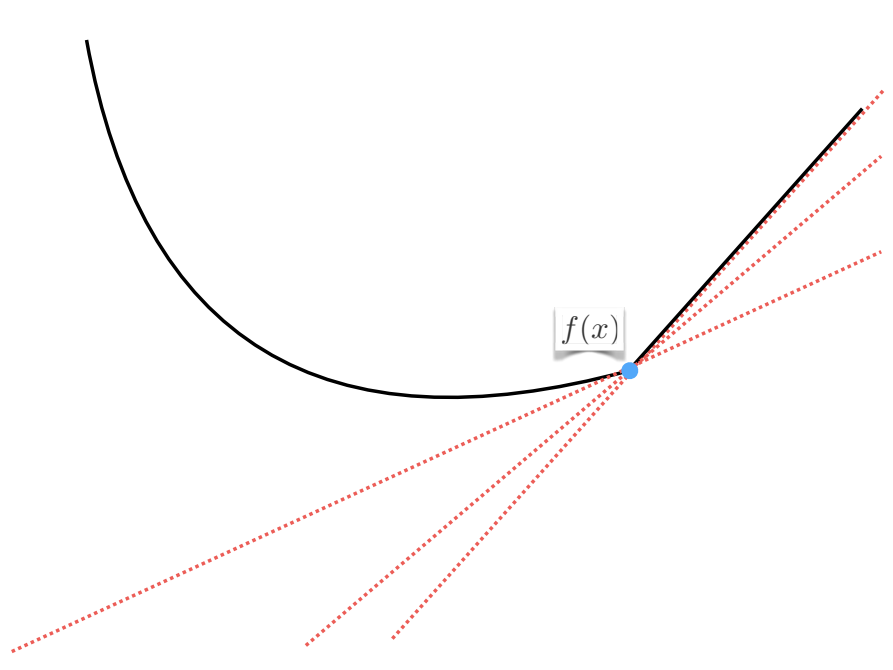
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

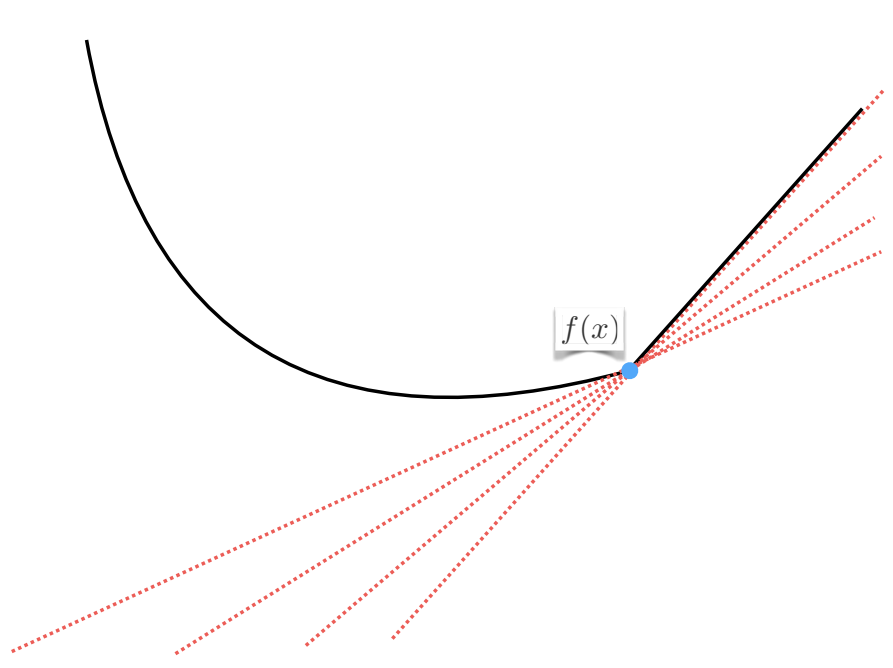
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

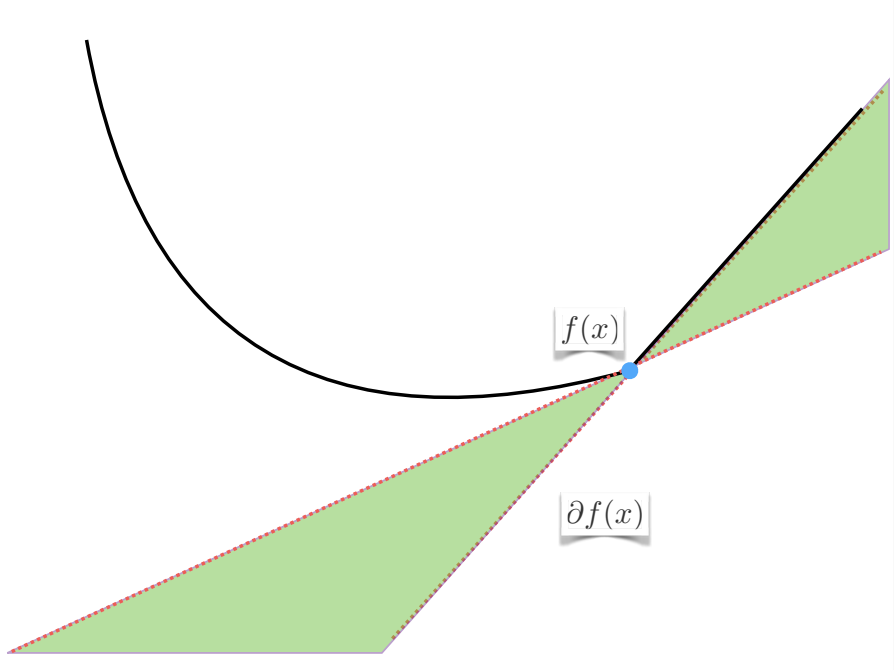
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Example

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

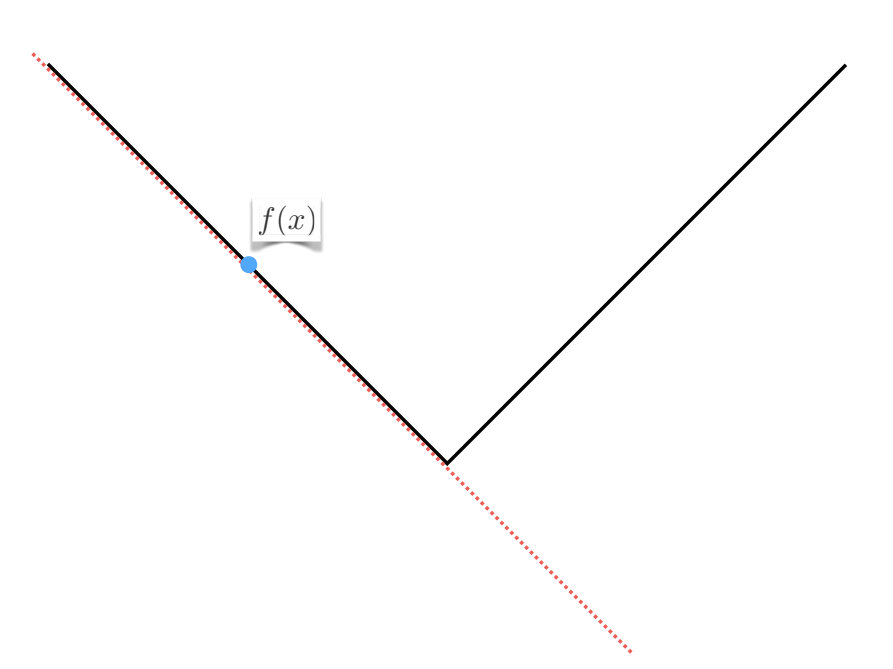
$$f(y) \geq f(x) + d^{\mathsf{T}}(y - x), \forall x, y.$$



Another example

Consider the absolute value function: $|x|$

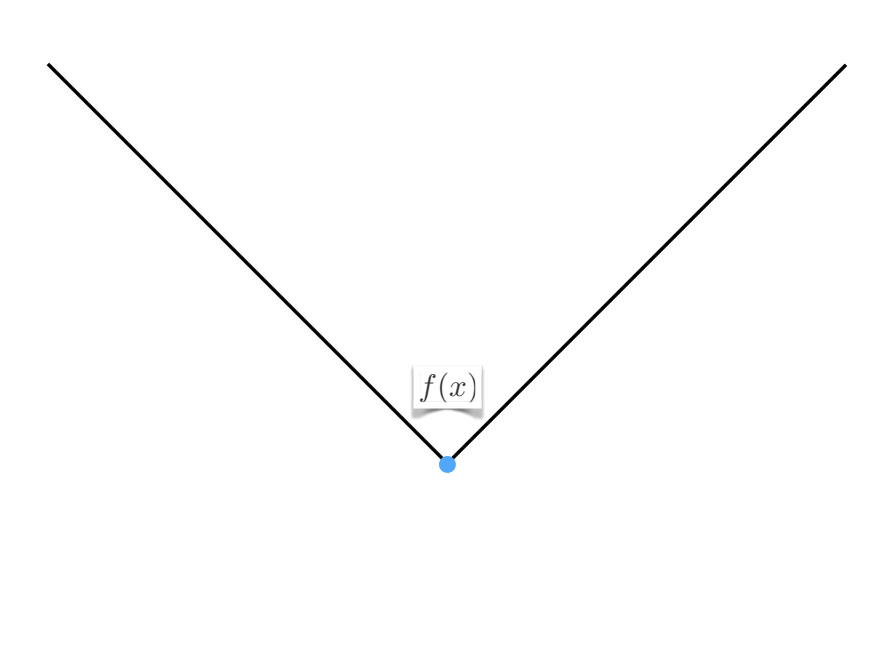
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

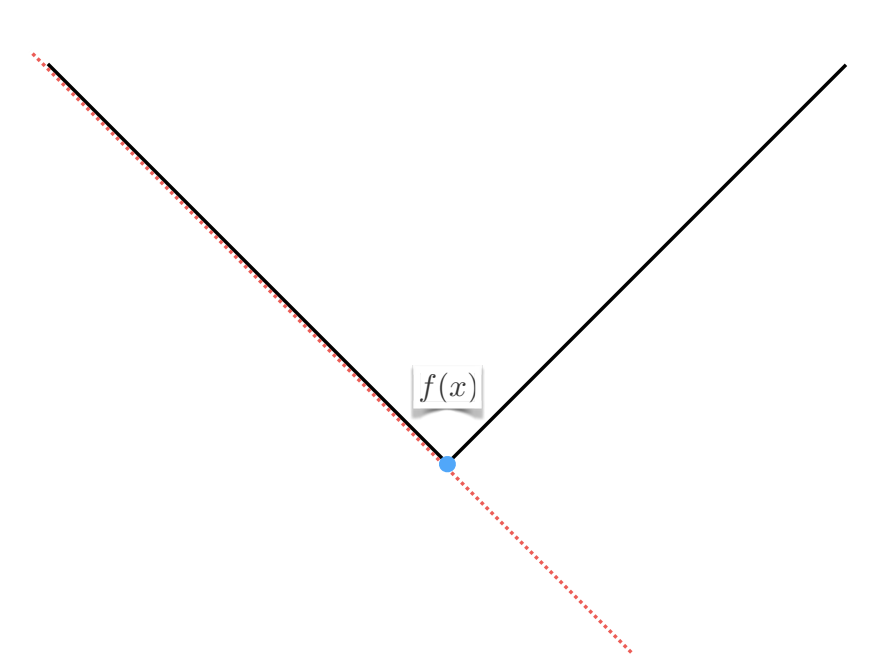
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

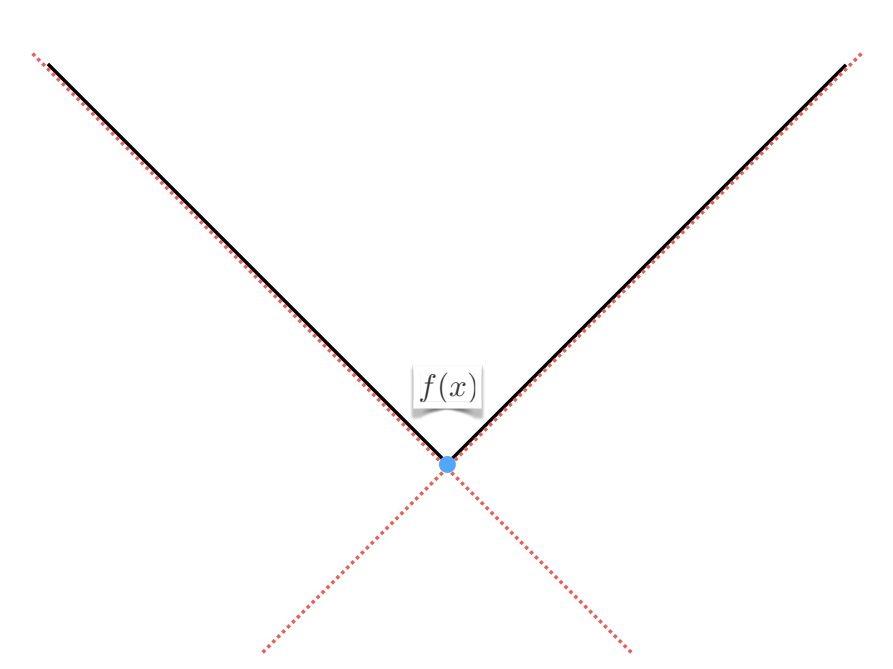
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

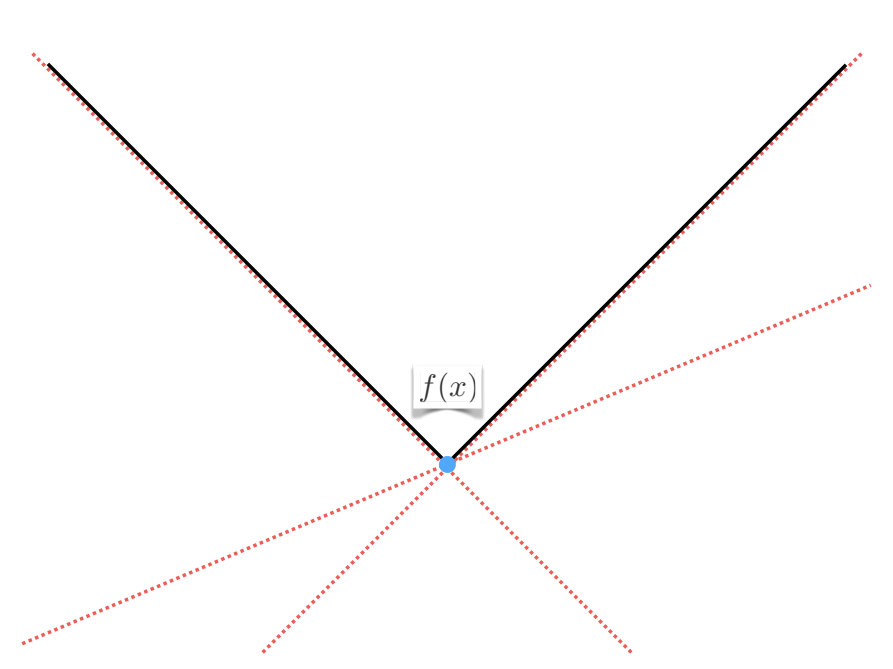
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

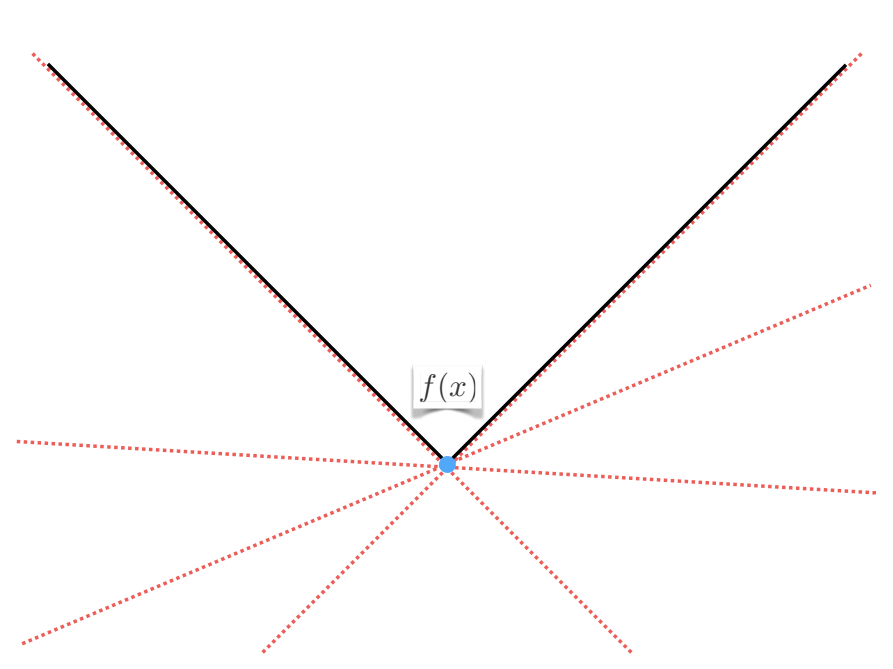
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

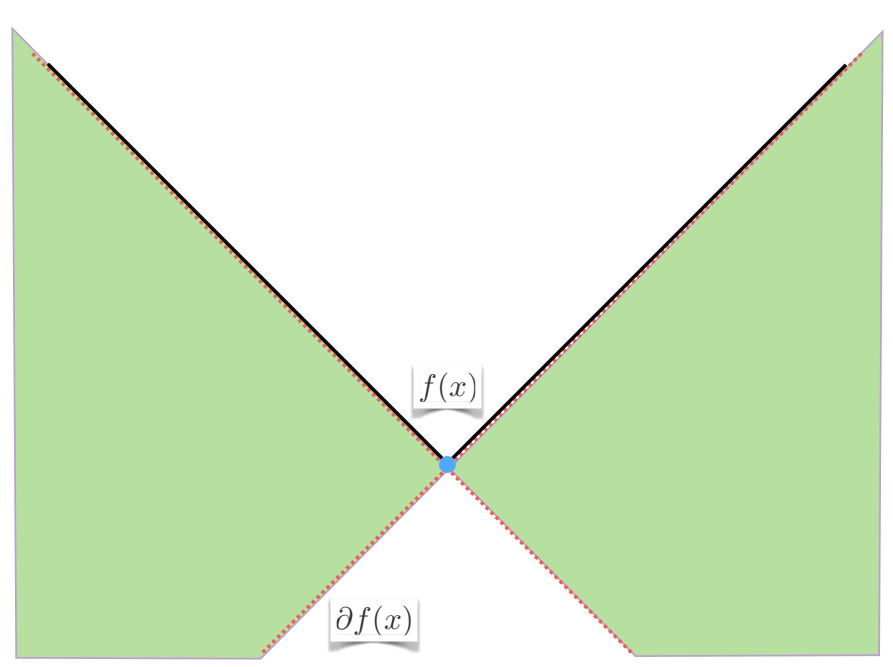
$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$



Another example

Consider the absolute value function: $|x|$

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



Subdifferential of max function

- $|x|$ is a special case of the max function
- Given two convex functions $f_1(x)$ and $f_2(x)$, the subdifferential of $\max(f_1(x), f_2(x))$ is given by

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

- I.e., any convex combination of the gradients of the *argmax*

Subdifferential of max function

- $|x|$ is a special case of the max function
- Given two convex functions $f_1(x)$ and $f_2(x)$, the subdifferential of $\max(f_1(x), f_2(x))$ is given by

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

- I.e., any convex combination of the gradients of the *argmax*

Subdifferential of max function

- $|x|$ is a special case of the max function
- Given two convex functions $f_1(x)$ and $f_2(x)$, the subdifferential of $\max(f_1(x), f_2(x))$ is given by

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

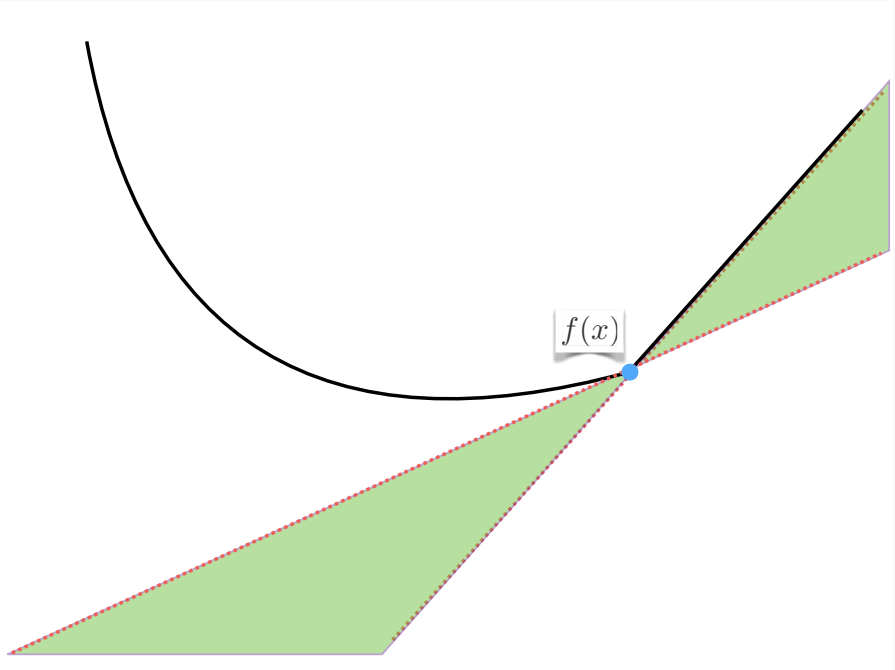
- I.e., any convex combination of the gradients of the *argmax*

The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general

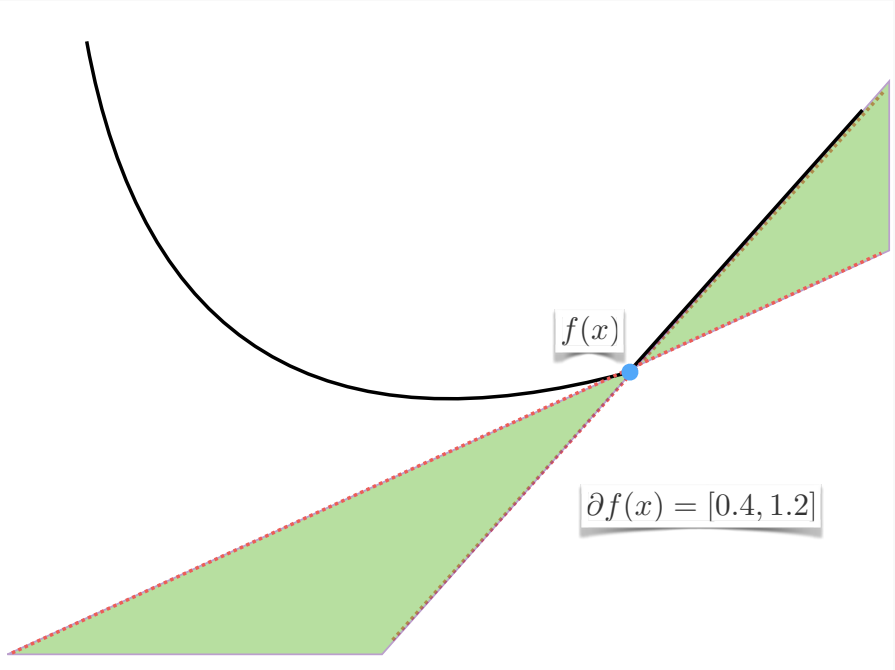


The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general

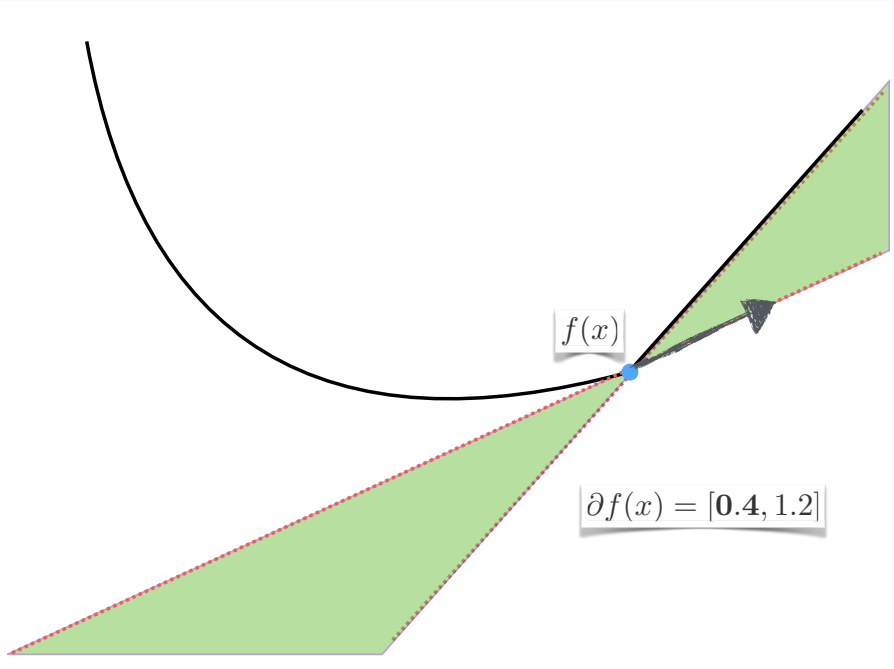


The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general

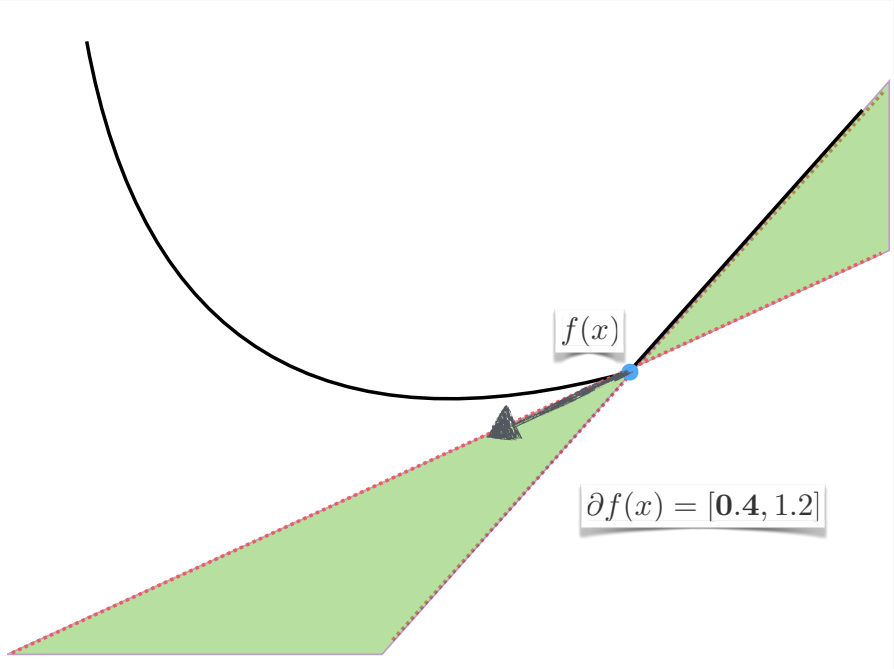


The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general



The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general
  - If $d_t \neq argmin_{\{} d \in \partial f(x)\} \|d\|$, the objective may increase
  - But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α
  - Again, for convergence, we require $\alpha \to 0$
- The basic stochastic subgradient method

$$x^{t+1} = x^t - \alpha_t d_{it}$$

  for some $d_{it} \in \partial f_{it}(x^t)$, $i_t \sim \{1, 2, \ldots, N\}$

The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general
  - If $d_t \neq argmin_{\{} d \in \partial f(x)\} \|d\|$, the objective may increase
  - But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α
  - Again, for convergence, we require $\alpha \to 0$
- The basic stochastic subgradient method

$$x^{t+1} = x^t - \alpha_t d_{it}$$

  for some $d_{it} \in \partial f_{it}(x^t)$, $i_t \sim \{1, 2, \ldots, N\}$

The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general
  - If $d_t \neq argmin_{\{} d \in \partial f(x)\} \|d\|$, the objective may increase
  - But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α
  - Again, for convergence, we require $\alpha \to 0$
- The basic stochastic subgradient method

$$x^{t+1} = x^t - \alpha_t d_{it}$$

  for some $d_{it} \in \partial f_{it}(x^t)$, $i_t \sim \{1, 2, \ldots, N\}$

The subgradient method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t d_t,$$

  for some $d_t \in \partial f(x^t)$
- The steepest descent d_t is $argmin_{d \in \partial f(x)} \{\|d\|\}$
  - Easy to see in the 1d case
  - Easy to find for ℓ_1 regularization, but hard in general
  - If $d_t \neq argmin_{\{} d \in \partial f(x)\} \|d\|$, the objective may increase
  - But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α
  - Again, for convergence, we require $\alpha \to 0$
- The basic stochastic subgradient method

$$x^{t+1} = x^t - \alpha_t d_{it}$$

  for some $d_{it} \in \partial f_{it}(x^t)$, $i_t \sim \{1, 2, \ldots, N\}$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

The stochastic subgradient method in practice

- Theory says we should do

$$i_t \sim \{1, 2, \ldots, N\}, \quad \alpha_t = \frac{1}{\mu_t}$$
$$x^{t+1} = x^t - \alpha \nabla f_{i_t}(x^t).$$

  - $\mathcal{O}(1/t)$ for smooth objectives
  - $\mathcal{O}(\log(t)/t)$ for non-smooth objectives
- Do not do this! Why?
  - Initial steps will be huge ($\mu_1 = 1/N$ or $1/\sqrt{N}$)
  - Later steps are tiny ($1/t$ get small very quickly)
  - Convergence rate is not robust to mis-specification of μ
  - Non-adaptive (very worst-case behaviour)
- What people do in practice
  - Use smaller initial steps, then go to zero more slowly
  - Take a weighted average of the iterations or gradients

$$\bar{x}_t = \sum_{i=1}^{t} w_t x_t, \qquad \bar{d}_t = \sum_{i=1}^{t} \delta_t d_t.$$

There is work that supports using large steps and averaging

- [Moulines and Bach, 2011], [Lacoste-Julien et al., 2012]
  - Averaging later iterations achieves $\mathcal{O}(1\ddot{)}$ in non-smooth case
  - Averaging by iteration number achieves the same
- [Nesterov, 2009], [Xiao, 2009]
  - Gradient averaging improves constants ('dual averaging')
  - Finds non-zero variables with sparse regularizers
- [Moulines and Bach, 2011]
  - $\alpha_t = \mathcal{O}(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = \mathcal{O}(1/t)$
- [Nedić and Bertsekas, 2001]
  - Constant step size ($\alpha_t = \alpha$) achieves rate of

$$\mathbb{E}\left[f\left(x^t\right)\right] - f(x^*) \leq (1 - 2\mu\alpha)^t \left(f\left(x^0\right) - f\left(x^*\right)\right) + \mathcal{O}(\alpha)$$

- [Polyak and Juditsky, 1992]
  - In the smooth case, iterate averaging is asymptotically optimal
  - Achieves same rate as optimal Stochastic Newton method

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

- What about accelerated/Newton-like stochastic methods?
  - Stochasticity in these methods does not improve the convergence rate
- But, it has been shown that
  - [Ghadimi and Lan, 2010]
    - Acceleration can improve dependence on L and μ
    - It improves performance at start if noise is small
  - Newton-line AdaGrad method [Duchi et al., 2011]

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

  - improves regret bounds, but not optimization error
  - Newton-like method [Bach and Moulines, 2013] achieves $\mathcal{O}(1/t)$ without strong-convexity (but with extra self-concordance assumption)

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

Recap

- We want to solve problems with BIG data $X \in \mathbb{R}^{D \times N}$
- When D is large, we use gradient methods
- When N is large, we use stochastic gradient methods
  - If the function is non-smooth, stochastic subgradient has great convergence rates
- Stochastic methods:
  - Are N times faster than deterministic methods
  - Do a lot of progress quickly, then stall
- In practice:
  - Choose smaller step sizes at the beginning
  - Averaging the iterations / gradients helps
  - Taking a permutation of the data (no longer unbiased gradient) works well too
- Next week Mohammed will talk about finite-sum methods

References

Bach, F. and Moulines, E. (2013).
Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n).
In *Advances in Neural Information Processing Systems*, pages 773–781.

Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
The Journal of Machine Learning Research, 12:2121–2159.

Ghadimi, S. and Lan, G. (2010).
Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization.
Optimization Online, July.

Lacoste-Julien, S., Schmidt, M., and Bach, F. (2012).
A simpler approach to obtaining an O(1/t) convergence rate for the projected stochastic subgradient method.
arXiv preprint arXiv:1212.2002.

Moulines, E. and Bach, F. R. (2011).
Non-asymptotic analysis of stochastic approximation algorithms for machine learning.
In *Advances in Neural Information Processing Systems*, pages 451–459.

Nedić, A. and Bertsekas, D. P. (2001).
Incremental subgradient methods for nondifferentiable optimization.
SIAM Journal on Optimization, 12(1):109–138.

Nesterov, Y. (2009).
Primal-dual subgradient methods for convex problems.
Mathematical programming, 120(1):221–259.

Polyak, B. T. and Juditsky, A. B. (1992).
Acceleration of stochastic approximation by averaging.
SIAM Journal on Control and Optimization, 30(4):838–855.

Robbins, H. and Monro, S. (1951).
A stochastic approximation method.
The annals of mathematical statistics, pages 400–407.

Tseng, P. (1998).
An incremental gradient (-projection) method with momentum term and adaptive stepsize rule.
SIAM Journal on Optimization, 8(2):506–531.

Xiao, L. (2009).
Dual averaging method for regularized stochastic learning and online optimization.
In *Advances in Neural Information Processing Systems*, pages 2116–2124.