Spring School - April 2016 - Spartan/Macsenet
Francis Bach
Slides generously provided by Mark Schmidt

Modern Convex Optimization Methods for Large-Scale Empirical Risk Minimization (Part I: Primal Methods)

International Conference on Machine Learning

Peter Richtárik and Mark Schmidt

July 2015

Further reading:

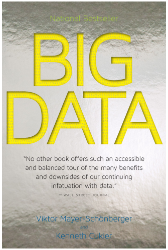
-Dimitri Bertsekas. Convex Optimization Algorithms, Athena Scientific, 2015.
-Yurii Nesterov. Introductory lectures on convex optimization: a basic course. Kluwer Academic Publishers, 2004.
-Sebastien Bubeck. Convex optimization: Algorithms and complexity. Foundations and Trends in Machine Learning, 8(3-4):231–357, 2015.

Context: Big Data and Big Models

- We are collecting data at unprecedented rates.
    - Seen across many fields of science and engineering.
    - Not gigabytes, but terabytes or petabytes (and beyond).

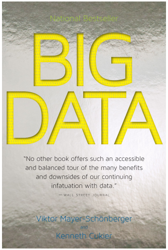
Context: Big Data and Big Models

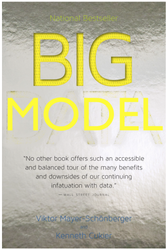
- We are collecting data at unprecedented rates.
  - Seen across many fields of science and engineering.
  - Not gigabytes, but terabytes or petabytes (and beyond).



Context: Big Data and Big Models

- We are collecting data at unprecedented rates.
  - Seen across many fields of science and engineering.
  - Not gigabytes, but terabytes or petabytes (and beyond).





- Machine learning can use big data to fit richer models:
  - Bioinformatics.
  - Computer vision.
  - Speech recognition.
  - Product recommendation.
  - Machine translation.

Common Framework: Empirical Risk Minimization

- The most common framework is empirical risk minimization:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

Common Framework: Empirical Risk Minimization

- The most common framework is empirical risk minimization:

$$\min_{x \in \mathbb{R}^P} \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(x, a_i, b_i)}_{\text{data fitting term}} \quad + \quad \underbrace{\lambda r(x)}_{\text{regularizer}}$$

  - We have n **observations** a_i (and possibly labels b_i).
  - We want to find optimal **parameters** x^*.

Common Framework: Empirical Risk Minimization

- The most common framework is empirical risk minimization:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

  - We have n **observations** a_i (and possibly labels b_i).
  - We want to find optimal **parameters** x^*.
- Examples range from squared error with 2-norm regularization,

$$\min_{x\in\mathbb{R}^P} \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}(a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2,$$

but also conditional random fields and deep neural networks.

Common Framework: Empirical Risk Minimization

- The most common framework is empirical risk minimization:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

  - We have n **observations** a_i (and possibly labels b_i).
  - We want to find optimal **parameters** x^*.
- Examples range from squared error with 2-norm regularization,

$$\min_{x \in \mathbb{R}^P} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2,$$

  but also conditional random fields and deep neural networks.
- Main practical challenges:
  - Designing/learning good features a_i.
  - Efficiently solving the problem when N or P are very large.

Motivation: Why Learn about Convex Optimization?

- Why learn about large-scale optimization?
  - Optimization is at the core of many ML algorithms.
  - Can't solve huge problems with traditional techniques.

Motivation: Why Learn about Convex Optimization?

- Why learn about large-scale optimization?
  - Optimization is at the core of many ML algorithms.
  - Can't solve huge problems with traditional techniques.
- Why in particular learn about convex optimization?

Motivation: Why Learn about Convex Optimization?

- Why learn about large-scale optimization?
  - Optimization is at the core of many ML algorithms.
  - Can't solve huge problems with traditional techniques.

- Why in particular learn about convex optimization?
  - Among only *efficiently-solvable* continuous problems.

Motivation: Why Learn about Convex Optimization?

- Why learn about large-scale optimization?
  - Optimization is at the core of many ML algorithms.
  - Can't solve huge problems with traditional techniques.
- Why in particular learn about convex optimization?
  - Among only *efficiently-solvable* continuous problems.
  - You can do a lot with convex models.

    (least squares, lasso, generlized linear models, SVMs, CRFs, etc.)
  - Empirically effective non-convex methods are often based methods with good properties for convex objectives.

    (functions are locally convex around minimizers)

Motivation: Why Learn about Convex Optimization?

- Why learn about large-scale optimization?
  - Optimization is at the core of many ML algorithms.
  - Can't solve huge problems with traditional techniques.

- Why in particular learn about convex optimization?
  - Among only *efficiently-solvable* continuous problems.
  - You can do a lot with convex models.

    (least squares, lasso, generlized linear models, SVMs, CRFs, etc.)
  - Empirically effective non-convex methods are often based methods with good properties for convex objectives.

    (functions are locally convex around minimizers)
  - Tools from convex analysis are being extended to non-convex.

How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$

How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

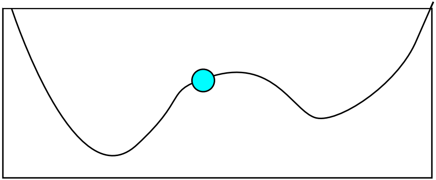
$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$



How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

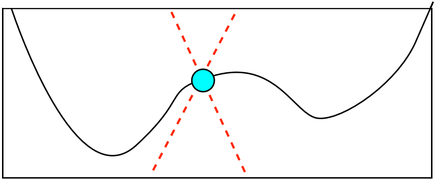
$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$



How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

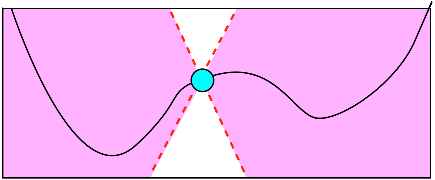
$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$



How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

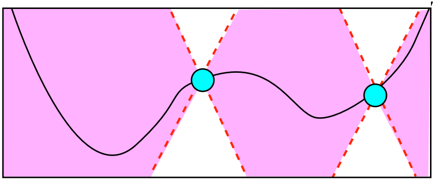
$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$



How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- After t iterations, the error of *any algorithm* is $\Omega(1/t^{1/n})$.

(and grid-search is nearly optimal)

How hard is real-valued optimization?

How long to find an ϵ-optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

- General function: impossible!

We need to make some assumptions about the function:

- Assume f is Lipschitz-continuous: (can not change too quickly)

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- After t iterations, the error of *any algorithm* is $\Omega(1/t^{1/n})$.

  (and grid-search is nearly optimal)

- Optimization is hard, but assumptions make a big difference.

  (we went from impossible to very slow)

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad for\ \theta \in [0,1].$$

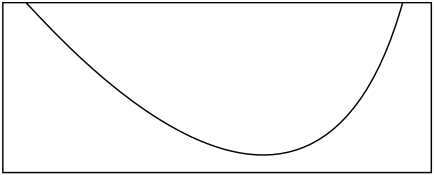
- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

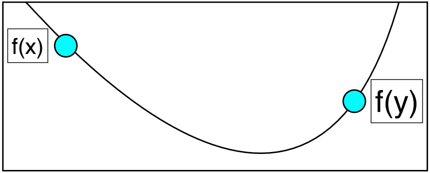


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

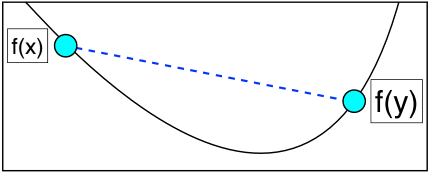


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

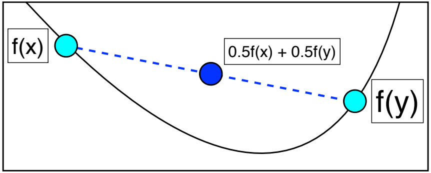


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

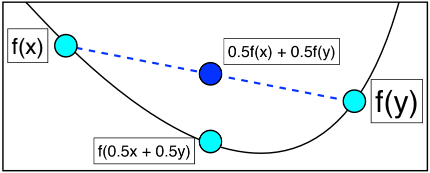


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

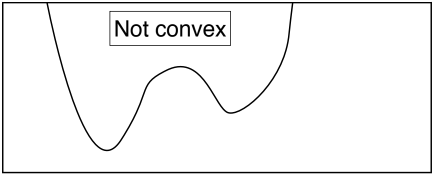


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.

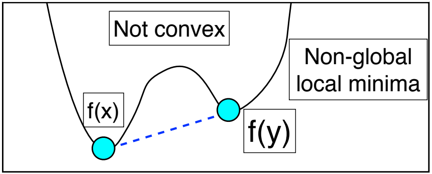


Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

- Function is below linear interpolation between x and y.
- Implies that all local minima are global minima.



Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad for\ \theta \in [0,1].$$

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \text{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

- The function is globally above the tangent at x.

Convex Functions: Three Characterizations

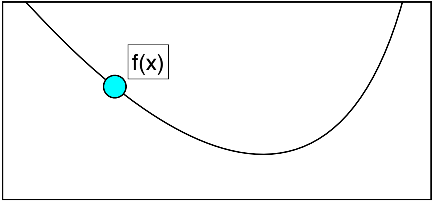
A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad for\ \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

- The function is globally above the tangent at x.



Convex Functions: Three Characterizations

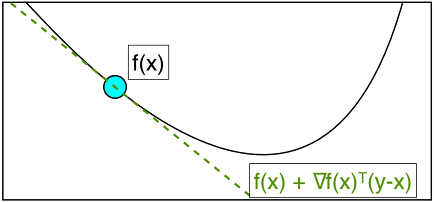
A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

- The function is globally above the tangent at x.



Convex Functions: Three Characterizations

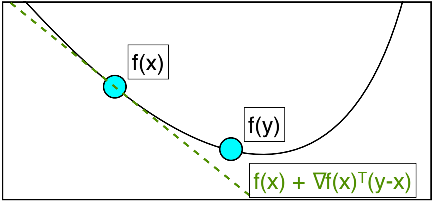
A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \text{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

- The function is globally above the tangent at x.



Convex Functions: Three Characterizations

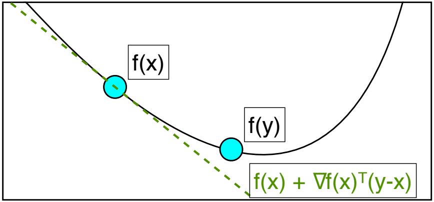
A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \text{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

- The function is globally above the tangent at x.



- If $\nabla f(y) = 0$, implies y is a a global minimizer.

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \text{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \textit{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

A twice-differentiable function f is convex if for all x we have

$$\nabla^2 f(x) \succeq 0$$

- All eigenvalues of 'Hessian' are non-negative.

Convex Functions: Three Characterizations

A function f is convex if for all x and y we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \text{for } \theta \in [0,1].$$

A differentiable function f is convex if for all x and y we have

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

A twice-differentiable function f is convex if for all x we have

$$\nabla^2 f(x) \succeq 0$$

- All eigenvalues of 'Hessian' are non-negative.
- The function is *flat or curved upwards* in every direction.
- This is usually the easiest way to show a function is convex.

Examples of Convex Functions

Some simple convex functions:

- $f(x) = c$
- $f(x) = a^T x$
- $f(x) = ax^2 + b$ (for $a > 0$)
- $f(x) = \exp(ax)$
- $f(x) = x \log x$ (for $x > 0$)
- $f(x) = \|x\|^2$
- $f(x) = \|x\|_p$
- $f(x) = \max_i \{x_i\}$

Examples of Convex Functions

Some simple convex functions:

- $f(x) = c$
- $f(x) = a^T x$
- $f(x) = ax^2 + b$ (for $a > 0$)
- $f(x) = \exp(ax)$
- $f(x) = x \log x$ (for $x > 0$)
- $f(x) = \|x\|^2$
- $f(x) = \|x\|_p$
- $f(x) = \max_i \{x_i\}$

Some other notable examples:

- $f(x, y) = \log(e^x + e^y)$
- $f(X) = \log \det X$ (for X positive-definite).
- $f(x, Y) = x^T Y^{-1} x$ (for Y positive-definite)

Operations that Preserve Convexity

1. Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x).$$

2. Composition with affine mapping:

$$g(x) = f(Ax + b).$$

3. Pointwise maximum:

$$f(x) = \max_i \{f_i(x)\}.$$

Operations that Preserve Convexity

1. Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x).$$

2. Composition with affine mapping:

$$g(x) = f(Ax + b).$$

3. Pointwise maximum:

$$f(x) = \max_i \{f_i(x)\}.$$

Show that least-residual problems are convex for any ℓ_p-norm:

$$f(x) = ||Ax - b||_p$$

Operations that Preserve Convexity

1. Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x).$$

2. Composition with affine mapping:

$$g(x) = f(Ax + b).$$

3. Pointwise maximum:

$$f(x) = \max_i \{f_i(x)\}.$$

Show that least-residual problems are convex for any ℓ_p-norm:

$$f(x) = ||Ax - b||_p$$

We know that $\| \cdot \|_p$ is a norm, so it follows from (2).

Operations that Preserve Convexity

1. Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x).$$

2. Composition with affine mapping:

$$g(x) = f(Ax + b).$$

3. Pointwise maximum:

$$f(x) = \max_i \{f_i(x)\}.$$

Show that SVMs are convex:

$$f(x) = \frac{1}{2}||x||^2 + C\sum_{i=1}^{n} \max\{0, 1 - b_i a_i^T x\}.$$

Operations that Preserve Convexity

1. Non-negative weighted sum:

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x).$$

2. Composition with affine mapping:

$$g(x) = f(Ax + b).$$

3. Pointwise maximum:

$$f(x) = \max_i \{f_i(x)\}.$$

Show that SVMs are convex:

$$f(x) = \frac{1}{2}||x||^2 + C\sum_{i=1}^{n} \max\{0, 1 - b_i a_i^T x\}.$$

The first term has Hessian $I \succ 0$, for the second term use (3) on the two (convex) arguments, then use (1) to put it all together.

Outline

Motivation for Gradient Methods

- We can solve convex optimization problems in polynomial-time by *interior-point* methods

Motivation for Gradient Methods

- We can solve convex optimization problems in polynomial-time by *interior-point* methods
- But these solvers require $O(P^2)$ or worse cost per iteration.
  - Infeasible for applications where P may be in the billions.

Motivation for Gradient Methods

- We can solve convex optimization problems in polynomial-time by *interior-point* methods
- But these solvers require $O(P^2)$ or worse cost per iteration.
  - Infeasible for applications where P may be in the billions.
- Large-scale problems have renewed interest gradient methods:

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

Motivation for Gradient Methods

- We can solve convex optimization problems in polynomial-time by *interior-point* methods
- But these solvers require $O(P^2)$ or worse cost per iteration.
  - Infeasible for applications where P may be in the billions.
- Large-scale problems have renewed interest gradient methods:

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

  - Only have $O(P)$ iteration cost!
  - But how many iterations are needed?

Logistic Regression with 2-Norm Regularization

- Let's consider logistic regression with 2-norm regularization:

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))) + \frac{\lambda}{2}\|x\|^2.$$

- Objective f is convex.
- First term is Lipschitz continuous, second term is not.

Logistic Regression with 2-Norm Regularization

- Let's consider logistic regression with 2-norm regularization:

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))) + \frac{\lambda}{2}\|x\|^2.$$

- Objective f is convex.
- First term is Lipschitz continuous, second term is not.
- But we have

$$\mu I \preceq \nabla^2 f(x) \preceq LI,$$

for some L and μ.

$(L \leq \frac{1}{4}\|A\|_2^2 + \lambda,\ \mu \geq \lambda)$

Logistic Regression with 2-Norm Regularization

- Let's consider logistic regression with 2-norm regularization:

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))) + \frac{\lambda}{2}\|x\|^2.$$

- Objective f is convex.
- First term is Lipschitz continuous, second term is not.
- But we have

$$\mu I \preceq \nabla^2 f(x) \preceq LI,$$

for some L and μ.

$(L \leq \frac{1}{4}\|A\|_2^2 + \lambda,\ \mu \geq \lambda)$

- We say that the gradient is Lipschitz-continuous.
- We say that the function is strongly-convex.

Properties of Lipschitz-Continuous Gradient

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x)$$

Properties of Lipschitz-Continuous Gradient

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*

Properties of Lipschitz-Continuous Gradient

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*
- Variant of gradient method if we set x^{t+1} to minimum y value:

$$x^{t+1} = x^t - \frac{1}{L}\nabla f(x^t).$$

- Plugging this value in:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2.$$

- Guaranteed decrease of objective.

Properties of Lipschitz-Continuous Gradient

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*

Properties of Lipschitz-Continuous Gradient

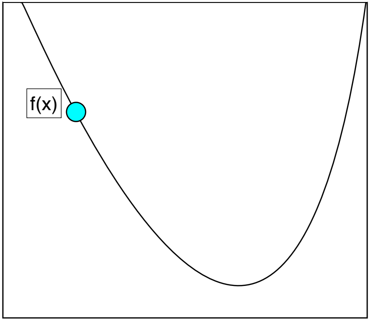
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*



Properties of Lipschitz-Continuous Gradient

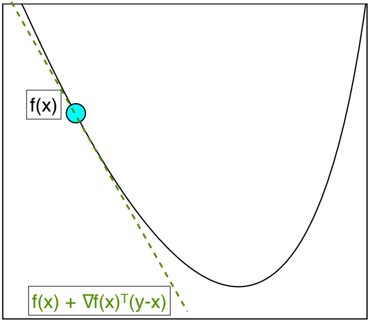
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*



Properties of Lipschitz-Continuous Gradient

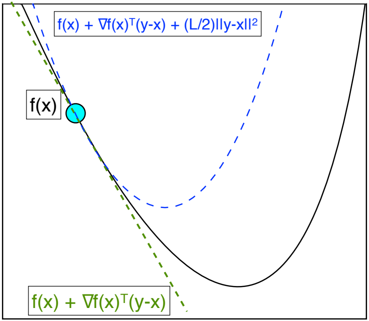
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*



Properties of Lipschitz-Continuous Gradient

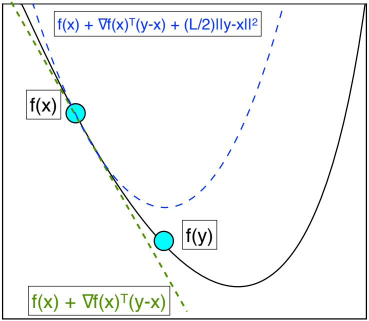
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*



Properties of Lipschitz-Continuous Gradient

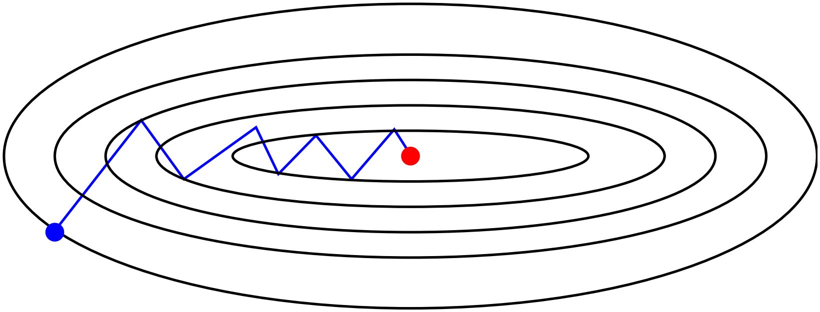
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \preceq LI$.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$

- *Global quadratic upper bound on function value.*



Properties of Strong-Convexity

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

Properties of Strong-Convexity

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \succeq \mu I$.

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

- *Global quadratic lower bound on function value.*

Properties of Strong-Convexity

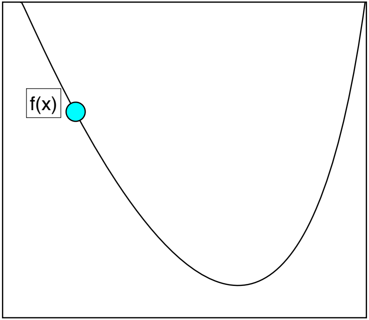
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x)$$

- Use that $\nabla^2 f(z) \succeq \mu I$.

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2$$

- *Global quadratic lower bound on function value.*



Properties of Strong-Convexity

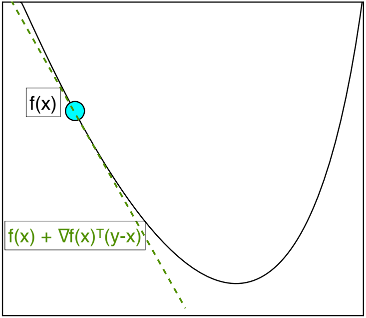
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \succeq \mu I$.

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

- *Global quadratic lower bound on function value.*



Properties of Strong-Convexity

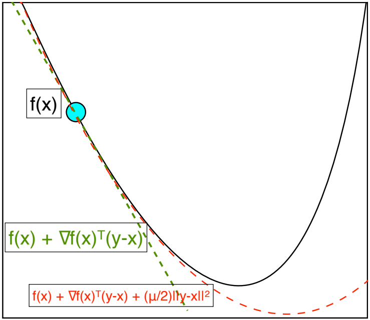
- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \succeq \mu I$.

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

- *Global quadratic lower bound on function value.*



Properties of Strong-Convexity

- From Taylor's theorem, for some z we have:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$

- Use that $\nabla^2 f(z) \succeq \mu I$.

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

- *Global quadratic lower bound on function value.*
- Minimize both sides in terms of y:

$$f(x^*) \geq f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2.$$

- Upper bound on how far we are from the solution.

Linear Convergence of Gradient Descent

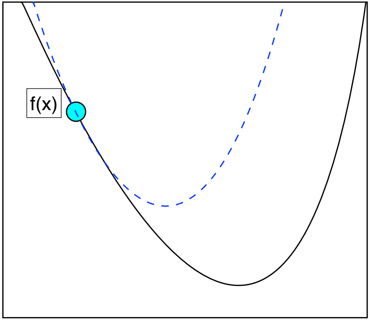
- We have bounds on x^{t+1} and x^*:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$

Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

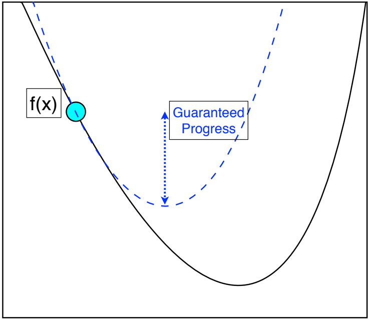
$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$



Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

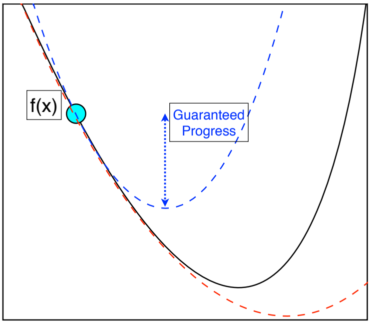
$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$



Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

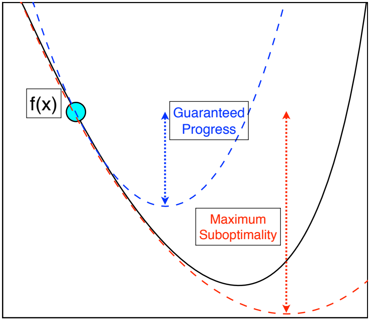
$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$



Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

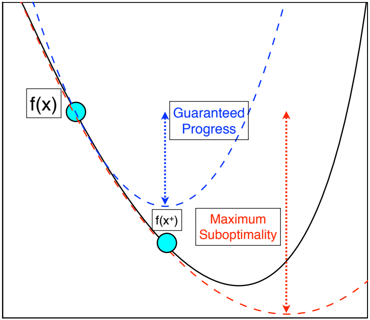
$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$



Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$



Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$

combine them to get

$$f(x^{t+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)[f(x^t) - f(x^*)]$$

Linear Convergence of Gradient Descent

- We have bounds on x^{t+1} and x^*:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2, \quad f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2.$$

combine them to get

$$f(x^{t+1}) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)[f(x^t) - f(x^*)]$$

- This gives a linear convergence rate:

$$f(x^t) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^t [f(x^0) - f(x^*)]$$

- Each iteration multiplies the error by a fixed amount.

(very fast if μ/L is not too close to one)

Maximum Likelihood Logistic Regression

- What about maximum-likelihood logistic regression?

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

Maximum Likelihood Logistic Regression

- What about maximum-likelihood logistic regression?

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$

Maximum Likelihood Logistic Regression

- What about maximum-likelihood logistic regression?

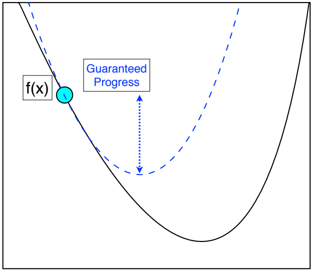
$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$



Maximum Likelihood Logistic Regression

- What about maximum-likelihood logistic regression?

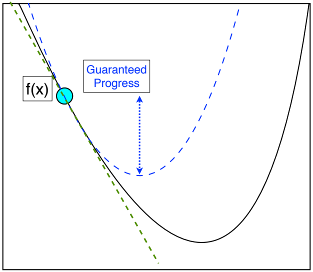
$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$



Maximum Likelihood Logistic Regression

- What about maximum-likelihood logistic regression?

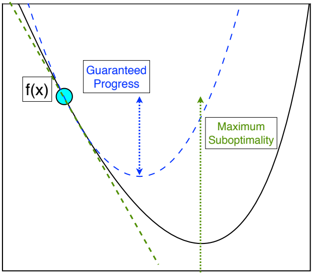
$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$



Maximum Likelihood Logistic Regression

- Consider maximum-likelihood logistic regression:

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$

- If some x^* exists, we have the sublinear convergence rate:

$$f(x^t) - f(x^*) = O(1/t)$$

(compare to slower $\Omega(1/t^{-1/N})$ for general Lipschitz functions)

Maximum Likelihood Logistic Regression

- Consider maximum-likelihood logistic regression:

$$f(x) = \sum_{i=1}^{n} \log(1 + exp(-b_i(x^T a_i))).$$

- We now only have

$$0 \preceq \nabla^2 f(x) \preceq LI.$$

- Convexity only gives a linear upper bound on $f(x^*)$:

$$f(x^*) \leq f(x) + \nabla f(x)^T (x^* - x)$$

- If some x^* exists, we have the sublinear convergence rate:

$$f(x^t) - f(x^*) = O(1/t)$$

(compare to slower $\Omega(1/t^{-1/N})$ for general Lipschitz functions)

- If f is convex, then $f + \lambda \|x\|^2$ is strongly-convex.

Gradient Method: Practical Issues

- In practice, searching for step size (line-search) is usually much faster than $\alpha = 1/L$.

(and doesn't require knowledge of L)

Gradient Method: Practical Issues

- In practice, searching for step size (line-search) is usually much faster than $\alpha = 1/L$.

(and doesn't require knowledge of L)

- Basic Armijo backtracking line-search:
  1. Start with a large value of α.
  2. Divide α in half until we satisfy (typically value is $\gamma = .0001$)

$$f(x^{t+1}) \leq f(x^t) - \gamma\alpha||\nabla f(x^t)||^2.$$

Gradient Method: Practical Issues

- In practice, searching for step size (line-search) is usually much faster than $\alpha = 1/L$.

(and doesn't require knowledge of L)

- Basic Armijo backtracking line-search:
  1. Start with a large value of α.
  2. Divide α in half until we satisfy (typically value is $\gamma = .0001$)

$$f(x^{t+1}) \leq f(x^t) - \gamma\alpha||\nabla f(x^t)||^2.$$

- Practical methods may use *Wolfe conditions* (so α isn't too small), and/or use *interpolation* to propose trial step sizes.

(with good interpolation, ≈ 1 evaluation of f per iteration)

Gradient Method: Practical Issues

- In practice, searching for step size (line-search) is usually much faster than $\alpha = 1/L$.

(and doesn't require knowledge of L)

- Basic Armijo backtracking line-search:
  1. Start with a large value of α.
  2. Divide α in half until we satisfy (typically value is $\gamma = .0001$)

$$f(x^{t+1}) \leq f(x^t) - \gamma\alpha||\nabla f(x^t)||^2.$$

- Practical methods may use *Wolfe conditions* (so α isn't too small), and/or use *interpolation* to propose trial step sizes.

(with good interpolation, ≈ 1 evaluation of f per iteration)

- Also, check your derivative code!

$$\nabla_i f(x) \approx \frac{f(x + \delta e_i) - f(x)}{\delta}$$

- For large-scale problems you can check a random direction d:

$$\nabla f(x)^T d \approx \frac{f(x + \delta d) - f(x)}{\delta}$$

Accelerated Gradient Method

- Is this the best algorithm under these assumptions?

Accelerated Gradient Method

- Is this the best algorithm under these assumptions?

Algorithm	Assumptions	Rate
Gradient	Convex	$O(1/t)$
Nesterov	Convex	$O(1/t^2)$
Gradient	Strongly-Convex	$O((1-\mu/L)^t)$
Nesterov	Strongly-Convex	$O((1-\sqrt{\mu/L})^t)$

Accelerated Gradient Method

- Is this the best algorithm under these assumptions?

Algorithm	Assumptions	Rate
Gradient	Convex	$O(1/t)$
Nesterov	Convex	$O(1/t^2)$
Gradient	Strongly-Convex	$O((1-\mu/L)^t)$
Nesterov	Strongly-Convex	$O((1-\sqrt{\mu/L})^t)$

- Nesterov's accelerated gradient method:

$$x_{t+1} = y_t - \alpha_t \nabla f(y_t),$$
$$y_{t+1} = x_t + \beta_t (x_{t+1} - x_t),$$

for appropriate α_t, β_t.

- Rate is nearly-optimal for dimension-independent algorithm.
- Similar to heavy-ball/momentum and conjugate gradient.

Accelerated Gradient Method

- Is this the best algorithm under these assumptions?

Algorithm	Assumptions	Rate
Gradient	Convex	$O(1/t)$
Nesterov	Convex	$O(1/t^2)$
Gradient	Strongly-Convex	$O((1-\mu/L)^t)$
Nesterov	Strongly-Convex	$O((1-\sqrt{\mu/L})^t)$

- Nesterov's accelerated gradient method:

$$x_{t+1} = y_t - \alpha_t \nabla f(y_t),$$
$$y_{t+1} = x_t + \beta_t(x_{t+1} - x_t),$$

  for appropriate α_t, β_t.

- Rate is nearly-optimal for dimension-independent algorithm.
- Similar to heavy-ball/momentum and conjugate gradient.
- For logistic regression and many other losses, we can get linear convergence without strong-convexity [Luo & Tseng, 1993].

Newton's Method

- The oldest differentiable optimization method is Newton's.

  (also called IRLS for functions of the form $f(Ax)$)

- Modern form uses the update

$$x^{t+1} = x^t - \alpha d,$$

where d is a solution to the system

$$\nabla^2 f(x) d = \nabla f(x).$$

(Assumes $\nabla^2 f(x) \succ 0$)

Newton's Method

- The oldest differentiable optimization method is Newton's.

(also called IRLS for functions of the form $f(Ax)$)

- Modern form uses the update

$$x^{t+1} = x^t - \alpha d,$$

where d is a solution to the system

$$\nabla^2 f(x) d = \nabla f(x).$$

(Assumes $\nabla^2 f(x) \succ 0$)

- Equivalent to minimizing the quadratic approximation:

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2\alpha} \|y - x\|^2_{\nabla^2 f(x)}.$$

(recall that $\|x\|_H^2 = x^T H x$)

Newton's Method

- The oldest differentiable optimization method is Newton's.

  (also called IRLS for functions of the form $f(Ax)$)

- Modern form uses the update

$$x^{t+1} = x^t - \alpha d,$$

where d is a solution to the system

$$\nabla^2 f(x) d = \nabla f(x).$$

(Assumes $\nabla^2 f(x) \succ 0$)

- Equivalent to minimizing the quadratic approximation:

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2\alpha} \|y - x\|^2_{\nabla^2 f(x)}.$$

(recall that $\|x\|^2_H = x^T H x$)

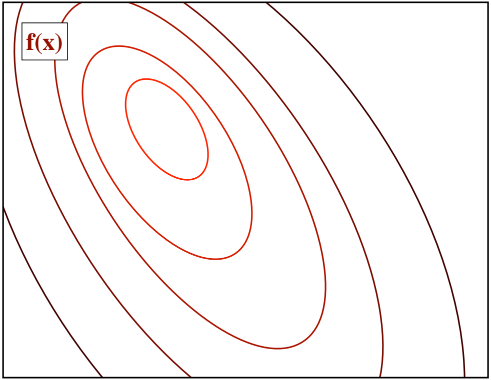
- We can generalize the Armijo condition to

$$f(x^{t+1}) \leq f(x^t) + \gamma \alpha \nabla f(x^t)^T d.$$

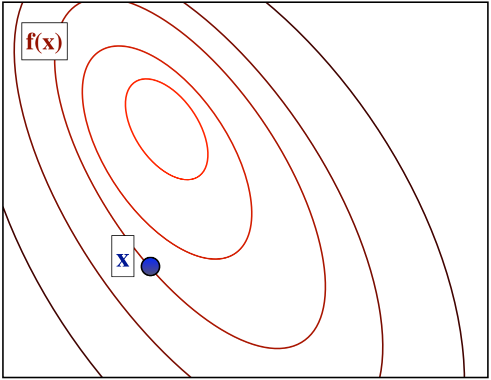
- Has a natural step length of $\alpha = 1$.

  (always accepted when close to a minimizer)

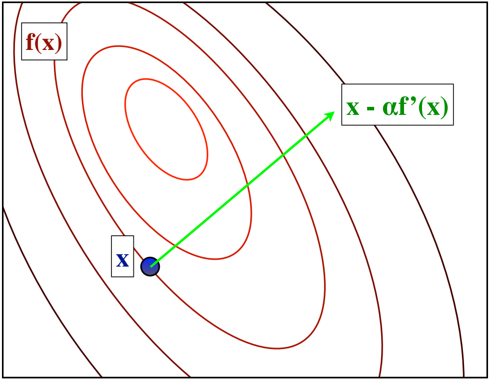
Newton's Method



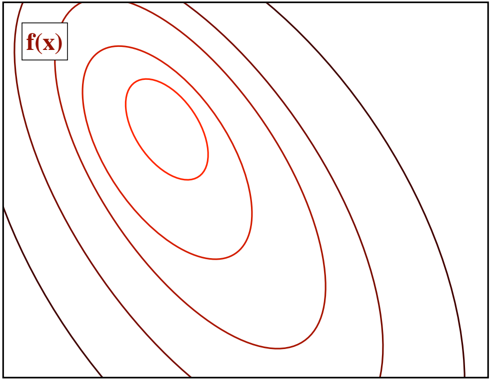
Newton's Method



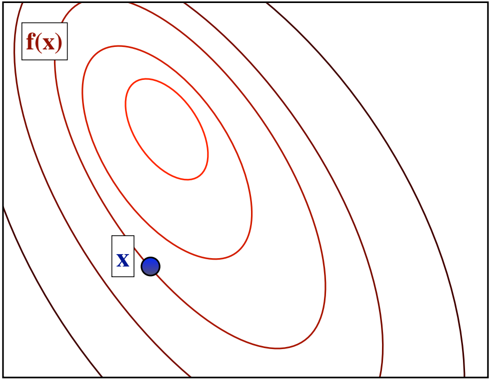
Newton's Method



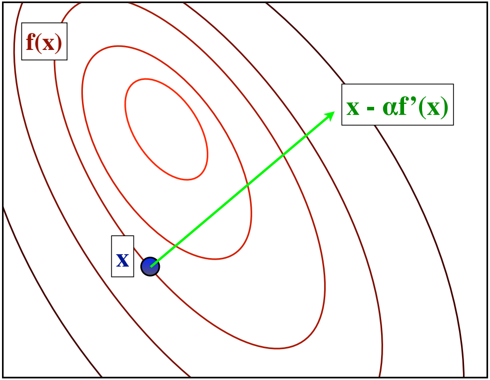
Newton's Method



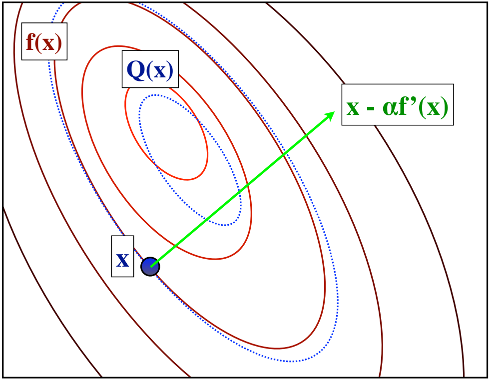
Newton's Method



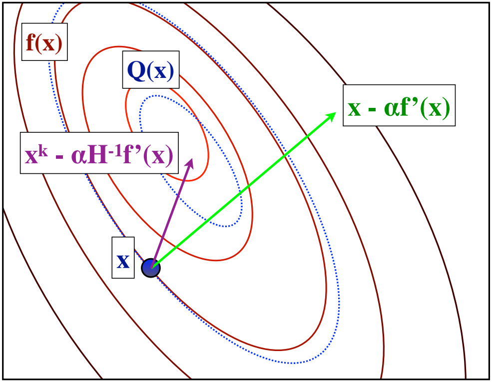
Newton's Method



Newton's Method



Newton's Method



Convergence Rate of Newton's Method

- If $\nabla^2 f(x)$ is Lipschitz-continuous and $\nabla^2 f(x) \succeq \mu$, then close to x^* Newton's method has local superlinear convergence:

$$f(x^{t+1}) - f(x^*) \leq \rho_t[f(x^t) - f(x^*)],$$

  with $\lim_{t\to\infty} \rho_t = 0$.
- Converges very fast, use it if you can!
- But requires solving $\nabla^2 f(x)d = \nabla f(x)$.

Convergence Rate of Newton's Method

- If $\nabla^2 f(x)$ is Lipschitz-continuous and $\nabla^2 f(x) \succeq \mu$, then close to x^* Newton's method has local superlinear convergence:

$$f(x^{t+1}) - f(x^*) \leq \rho_t[f(x^t) - f(x^*)],$$

  with $\lim_{t \to \infty} \rho_t = 0$.
- Converges very fast, use it if you can!
- But requires solving $\nabla^2 f(x) d = \nabla f(x)$.
- Get global rates under various assumptions (cubic-regularization/accelerated/self-concordant).

Newton's Method: Practical Issues

There are many practical variants of Newton's method:

- Modify the Hessian to be positive-definite.
- Only compute the Hessian every m iterations.
- Only use the diagonals of the Hessian.
- Quasi-Newton: Update a (diagonal plus low-rank) approximation of the Hessian (BFGS, L-BFGS).

Newton's Method: Practical Issues

There are many practical variants of Newton's method:

- Modify the Hessian to be positive-definite.
- Only compute the Hessian every m iterations.
- Only use the diagonals of the Hessian.
- Quasi-Newton: Update a (diagonal plus low-rank) approximation of the Hessian (BFGS, L-BFGS).
- Hessian-free: Compute d inexactly using Hessian-vector products:

$$\nabla^2 f(x)d = \lim_{\delta \to 0} \frac{\nabla f(x + \delta d) - \nabla f(x)}{\delta}$$

- Barzilai-Borwein: Choose a step-size that acts like the Hessian over the last iteration:

$$\alpha = \frac{(x^{t+1} - x^t)^T(\nabla f(x^{t+1}) - \nabla f(x^t))}{\|\nabla f(x^{t+1}) - f(x^t)\|^2}$$

Another related method is nonlinear conjugate gradient.

Outline

Big-N Problems

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

Big-N Problems

- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^P} \underbrace{\frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i)}_{\text{data fitting term}} \quad + \quad \underbrace{\lambda r(x)}_{\text{regularizer}}$$

- What if number of training examples N is very large?
  - E.g., ImageNet has more than 14 million annotated images.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$.

Stochastic vs. Deterministic Gradient Methods

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

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

  - Iteration cost is linear in N.
  - Convergence with constant α_t or line-search.

Stochastic vs. Deterministic Gradient Methods

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

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

  - Iteration cost is linear in N.
  - Convergence with constant α_t or line-search.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i from $\{1, 2, \ldots, N\}$.

$$x_{t+1} = x_t - \alpha_t f_i'(x_t).$$

Stochastic vs. Deterministic Gradient Methods

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

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

  - Iteration cost is linear in N.
  - Convergence with constant α_t or line-search.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i from $\{1, 2, \ldots, N\}$.

$$x_{t+1} = x_t - \alpha_t f_i'(x_t).$$

  - Gives unbiased estimate of true gradient,

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

  - Iteration cost is independent of N.

Stochastic vs. Deterministic Gradient Methods

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

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

  - Iteration cost is linear in N.
  - Convergence with constant α_t or line-search.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i from $\{1, 2, \dots, N\}$.

$$x_{t+1} = x_t - \alpha_t f_i'(x_t).$$

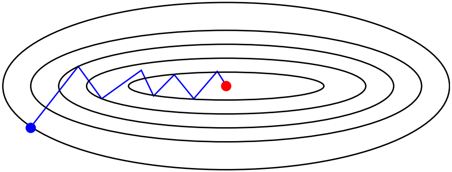
  - Gives unbiased estimate of true gradient,

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

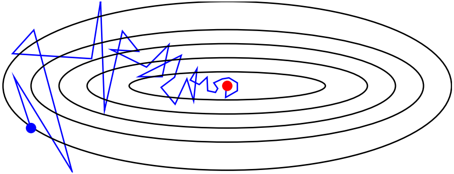
  - Iteration cost is independent of N.
  - Convergence requires $\alpha_t \to 0$.

Stochastic vs. Deterministic Gradient Methods

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



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



Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are N times faster, but how many iterations?

Stochastic vs. Deterministic Gradient Methods

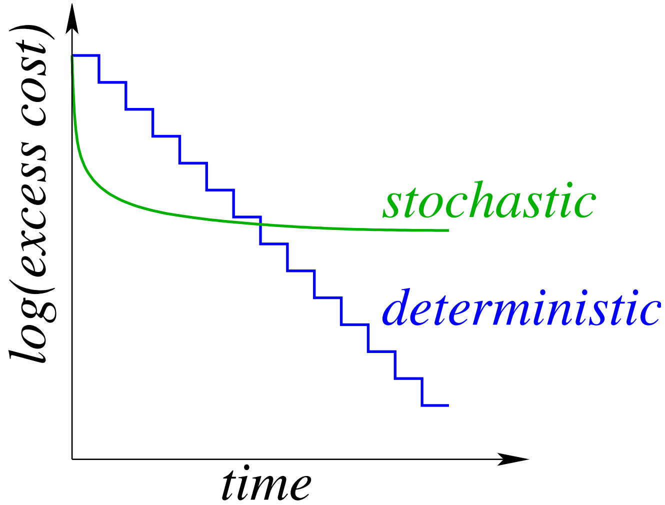
Stochastic iterations are N times faster, but how many iterations?

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

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

Stochastic vs. Deterministic Convergence Rates

Plot of convergence rates in strongly-convex case:



Stochastic will be superior for low-accuracy/time situations.

Stochastic vs. Deterministic for Non-Smooth

- Consider the binary support vector machine objective:

$$f(x) = \sum_{i=1}^{n} \max\{0, 1 - b_i(x^T a_i)\} + \frac{\lambda}{2}\|x\|^2.$$

Stochastic vs. Deterministic for Non-Smooth

- Consider the binary support vector machine objective:

$$f(x) = \sum_{i=1}^{n} \max\{0, 1 - b_i(x^T a_i)\} + \frac{\lambda}{2}\|x\|^2.$$

- Rates for subgradient methods for non-smooth objectives:

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

- Other black-box methods (cutting plane) are not faster.

Stochastic vs. Deterministic for Non-Smooth

- Consider the binary support vector machine objective:

$$f(x) = \sum_{i=1}^{n} \max\{0, 1 - b_i(x^T a_i)\} + \frac{\lambda}{2}\|x\|^2.$$

- Rates for subgradient methods for non-smooth objectives:

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

- Other black-box methods (cutting plane) are not faster.
- For non-smooth problems:
  - Stochastic methods have same rate as smooth case.
  - Deterministic methods are not faster than stochastic method.
  - So use stochastic subgradient (iterations are n times faster).

Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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

Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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

- At differentiable x:
  - Only subgradient is $\nabla f(x)$.
- At non-differentiable x:
  - We have a set of subgradients.
  - Called the sub-differential, $\partial f(x)$.
- Note that $0 \in \partial f(x)$ iff x is a global minimum.

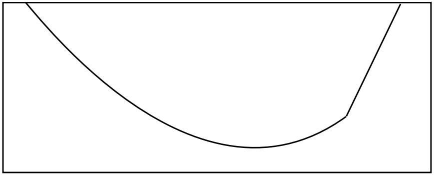
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



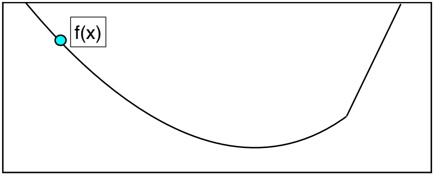
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



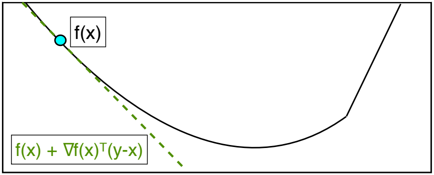
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



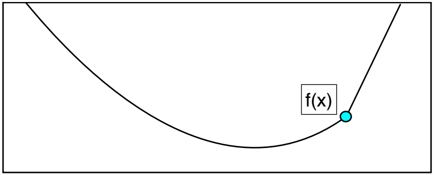
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



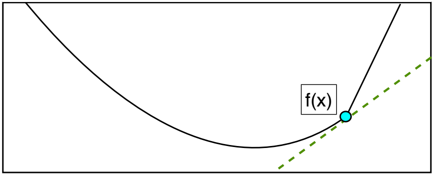
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



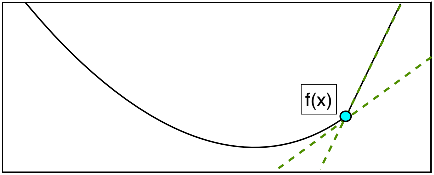
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



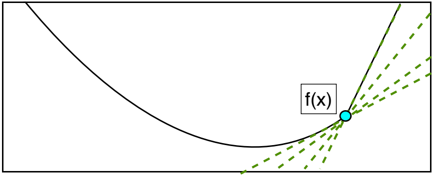
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



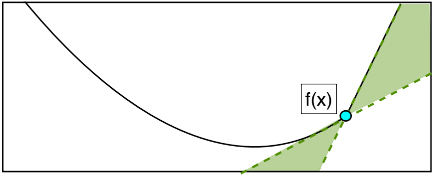
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

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

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

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



Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

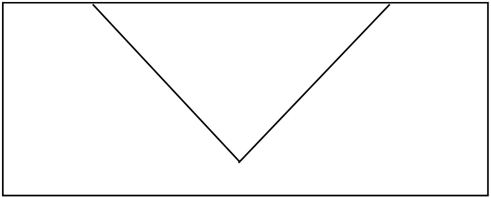
(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

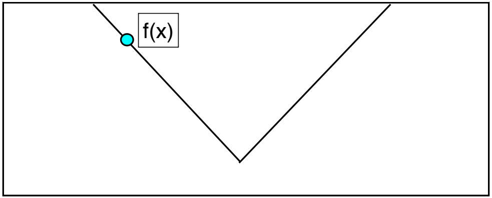


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

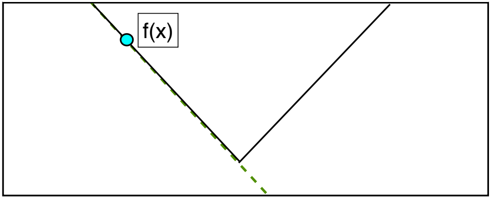


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

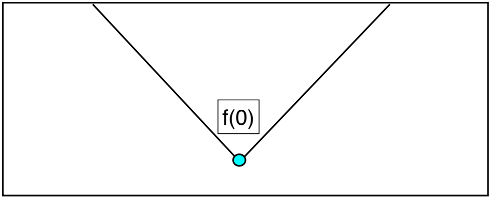


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

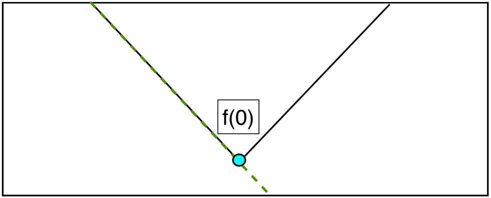


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

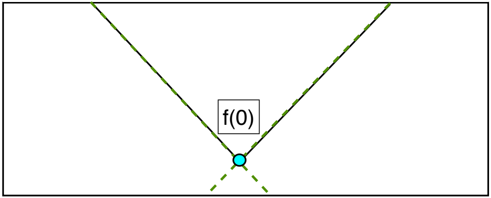


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

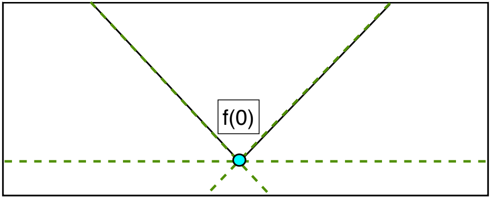


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

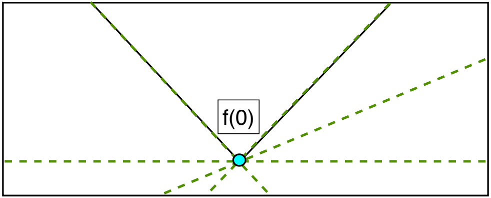


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

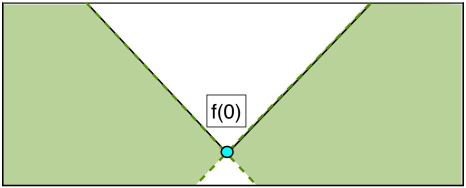


Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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



Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

- Sub-differential of max function:

$$\partial \max\{f_1(x), f_2(x)\} =$$

Sub-Differential of Absolute Value and Max Functions

- Sub-differential of absolute value function:

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

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

- Sub-differential of max function:

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

(any convex combination of the gradients of the argmax)

Subgradient and Stochastic Subgradient methods

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha d,$$

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

Subgradient and Stochastic Subgradient methods

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha d,$$

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

- The *steepest descent* choice is given by $\text{argmin}_{d \in \partial f(x)} \{\|d\|\}$.

  (often hard to compute, but easy for ℓ_1-regularization)

Subgradient and Stochastic Subgradient methods

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha d,$$

  for some $d \in \partial f(x^t)$.
- The *steepest descent* choice is given by $\text{argmin}_{d \in \partial f(x)}\{\|d\|\}$.
  (often hard to compute, but easy for ℓ_1-regularization)
- Otherwise, may increase the objective even for small α.
- But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α.
- For convergence, we require $\alpha \to 0$.

Subgradient and Stochastic Subgradient methods

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha d,$$

  for some $d \in \partial f(x^t)$.
- The *steepest descent* choice is given by $\text{argmin}_{d \in \partial f(x)}\{\|d\|\}$.
  (often hard to compute, but easy for ℓ_1-regularization)
- Otherwise, may increase the objective even for small α.
- But $\|x^{t+1} - x^*\| \leq \|x^t - x^*\|$ for small enough α.
- For convergence, we require $\alpha \to 0$.
- The basic stochastic subgradient method:

$$x^{t+1} = x^t - \alpha d,$$

  for some $d \in \partial f_i(x^t)$ for some random $i \in \{1, 2, \ldots, N\}$.

Stochastic Subgradient Methods in Practice

- The theory says to use decreasing sequence $\alpha_t = 1/\mu t$:

$$i_t = \mathrm{rand}(1, 2, \ldots, N), \quad \alpha_t = \frac{1}{\mu t}$$

$$x^{t+1} = x^t - \alpha_t f'_{i_t}(x^t).$$

  - $O(1/t)$ for smooth objectives.
  - $O(\log(t)/t)$ for non-smooth objectives.

Stochastic Subgradient Methods in Practice

- The theory says to use decreasing sequence $\alpha_t = 1/\mu t$:

$$i_t = \mathrm{rand}(1, 2, \ldots, N), \quad \alpha_t = \frac{1}{\mu t}$$

$$x^{t+1} = x^t - \alpha_t f'_{i_t}(x^t).$$

  - $O(1/t)$ for smooth objectives.
  - $O(\log(t)/t)$ for non-smooth objectives.
- Except for some special cases, you should not do this.
  - Initial steps are huge: usually $\mu = O(1/N)$ or $O(1/\sqrt{N})$.
  - Later steps are tiny: $1/t$ gets small very quickly.
  - Convergence rate is not robust to mis-specification of μ.
  - No adaptation to 'easier' problems than worst case.

Stochastic Subgradient Methods in Practice

- The theory says to use decreasing sequence $\alpha_t = 1/\mu t$:

$$i_t = \mathrm{rand}(1, 2, \dots, N), \quad \alpha_t = \frac{1}{\mu t}$$

$$x^{t+1} = x^t - \alpha_t f'_{i_t}(x^t).$$

  - $O(1/t)$ for smooth objectives.
  - $O(\log(t)/t)$ for non-smooth objectives.
- Except for some special cases, you should not do this.
  - Initial steps are huge: usually $\mu = O(1/N)$ or $O(1/\sqrt{N})$.
  - Later steps are tiny: $1/t$ gets small very quickly.
  - Convergence rate is not robust to mis-specification of μ.
  - No adaptation to 'easier' problems than worst case.
- Tricks that can improve theoretical and practical properties:
  1. Use smaller initial step-sizes, that go to zero more slowly.
  2. Take a (weighted) average of the iterations or gradients:

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

Speeding up Stochastic Subgradient Methods

Works that support using large steps and averaging:

- Rakhlin et at. [2011]:
  - Averaging later iterations achieves $O(1/t)$ in non-smooth case.
- Nesterov [2007], Xiao [2010]:
  - Gradient averaging improves constants ('dual averaging').
  - Finds non-zero variables with sparse regularizers.
- Bach & Moulines [2011]:
  - $\alpha_t = O(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = O(1/t)$.

Speeding up Stochastic Subgradient Methods

Works that support using large steps and averaging:

- Rakhlin et at. [2011]:
  - Averaging later iterations achieves $O(1/t)$ in non-smooth case.
- Nesterov [2007], Xiao [2010]:
  - Gradient averaging improves constants ('dual averaging').
  - Finds non-zero variables with sparse regularizers.
- Bach & Moulines [2011]:
  - $\alpha_t = O(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = O(1/t)$.
- Nedic & Bertsekas [2000]:
  - Constant step size ($\alpha_t = \alpha$) achieves rate of

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

Speeding up Stochastic Subgradient Methods

Works that support using large steps and averaging:

- Rakhlin et at. [2011]:
  - Averaging later iterations achieves $O(1/t)$ in non-smooth case.
- Nesterov [2007], Xiao [2010]:
  - Gradient averaging improves constants ('dual averaging').
  - Finds non-zero variables with sparse regularizers.
- Bach & Moulines [2011]:
  - $\alpha_t = O(1/t^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_t = O(1/t)$.
- Nedic & Bertsekas [2000]:
  - Constant step size ($\alpha_t = \alpha$) achieves rate of

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

- Polyak & Juditsky [1992]:
  - In smooth case, iterate averaging is asymptotically optimal.
  - Achieves same rate as optimal stochastic Newton method.

Stochastic Newton Methods?

- Should we use accelerated/Newton-like stochastic methods?
    - These do not improve the convergence rate.

Stochastic Newton Methods?

- Should we use accelerated/Newton-like stochastic methods?
  - These do not improve the convergence rate.
- But some positive results exist.
  - Ghadimi & Lan [2010]:
    - Acceleration can improve dependence on L and μ.
    - Improves performance at start or if noise is small.
  - Duchi et al. [2010]:
    - Newton-like methods can improve regret bounds.
  - Bach & Moulines [2013]:
    - Newton-like method achieves $O(1/t)$ without strong-convexity.
      (under extra self-concordance assumption)

Outline

Big-N Problems

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

Big-N Problems

- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^P} \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(x, a_i, b_i)}_{\text{data fitting term}} \quad + \quad \underbrace{\lambda r(x)}_{\text{regularizer}}$$

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

Big-N Problems

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Stochastic methods:
  - $O(1/t)$ convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
- Deterministic methods:
  - $O(\rho^t)$ convergence but requires N gradients per iteration.
  - The faster rate is possible because N is finite.

Big-N Problems

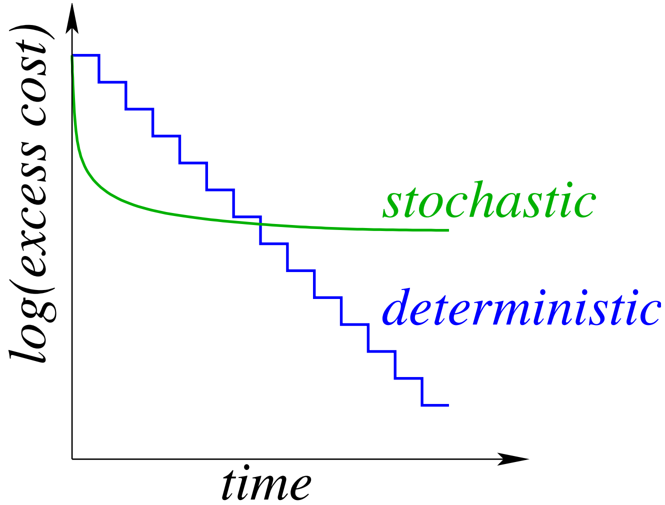
- Recall the regularized empirical risk minimization problem:

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

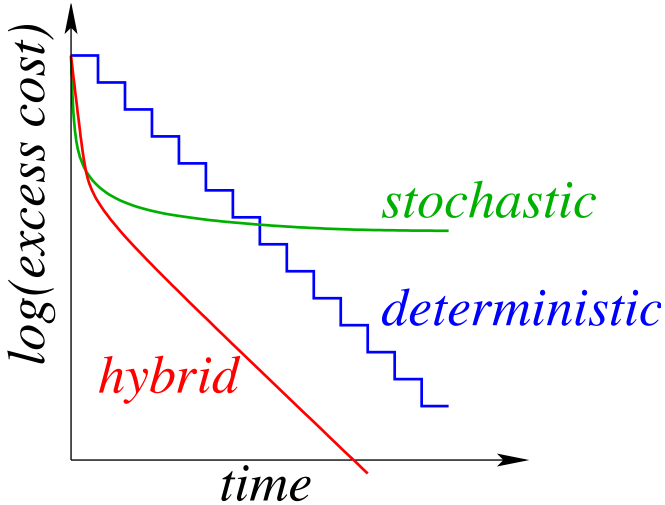
$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Stochastic methods:
  - $O(1/t)$ convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
- Deterministic methods:
  - $O(\rho^t)$ convergence but requires N gradients per iteration.
  - The faster rate is possible because N is finite.
- For minimizing finite sums, can we design a better method?

Motivation for Hybrid Methods



Motivation for Hybrid Methods



Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.

Hybrid Deterministic-Stochastic

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

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

- The SG method approximates it with 1 sample,

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

Hybrid Deterministic-Stochastic

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

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

- The SG method approximates it with 1 sample,

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

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

$$\frac{1}{|\mathcal{B}^t|}\sum_{i \in \mathcal{B}^t} f_i'(x^t) \approx \frac{1}{N}\sum_{i=1}^{N} f_i(x^t).$$

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

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

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

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

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

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|\mathcal{B}^t|$ increases**.

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

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

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|\mathcal{B}^t|$ increases.**
- Common to gradually increase the sample size $|\mathcal{B}^t|$.
  [Bertsekas & Tsitsiklis, 1996]

Approach 1: Batching

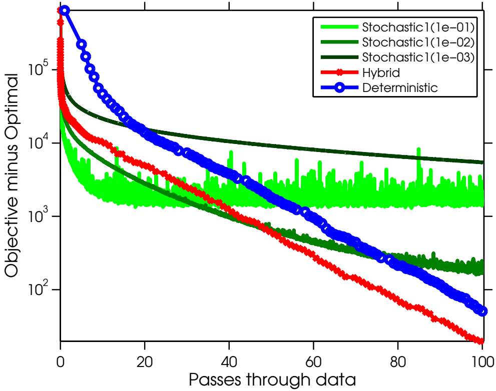
- The SG method with a sample $\mathcal{B}^t$ uses iterations

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

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|\mathcal{B}^t|$ increases.**
- Common to gradually increase the sample size $|\mathcal{B}^t|$.
  [Bertsekas & Tsitsiklis, 1996]
- We can choose $|\mathcal{B}^t|$ to achieve a linear convergence rate:
  - Early iterations are cheap like SG iterations.
  - Later iterations can use a Newton-like method.

Evaluation on Chain-Structured CRFs

Results on chain-structured conditional random field:



Stochastic Average Gradient

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

Stochastic Average Gradient

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

Stochastic Average Gradient

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

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} \nabla f_i(x^t)$$

Stochastic Average Gradient

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

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} \nabla f_i(x^t)$$

Stochastic Average Gradient

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

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

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

Stochastic Average Gradient

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

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

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected. [Le Roux et al., 2012]
  - Stochastic variant of increment average gradient (IAG). [Blatt et al., 2007]

Stochastic Average Gradient

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

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

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected. [Le Roux et al., 2012]
  - Stochastic variant of increment average gradient (IAG). [Blatt et al., 2007]
  - Assumes gradients of non-selected examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.

Convergence Rate of SAG

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

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

where

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

Convergence Rate of SAG

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

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

where

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

- Linear convergence rate but only 1 gradient per iteration.
    - For well-conditioned problems, constant reduction per pass:

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

    - For ill-conditioned problems, almost same as deterministic method (but N times faster).

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.

Rate of Convergence Comparison

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

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

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

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.
- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).
- Number of f_i' evaluations to reach ϵ:

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

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

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

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

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

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

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

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

Rate of Convergence Comparison

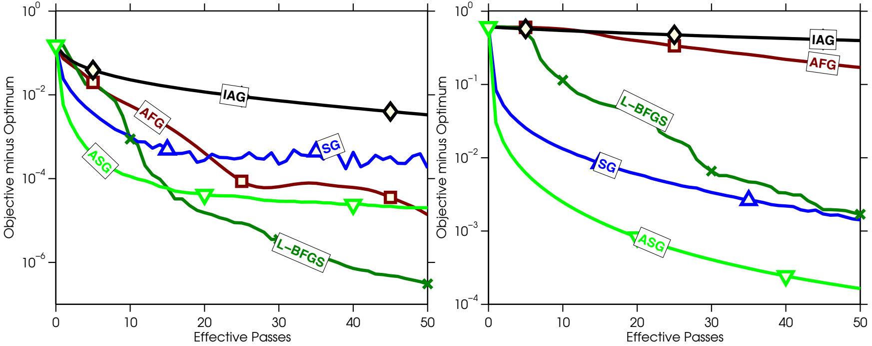
- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

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

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

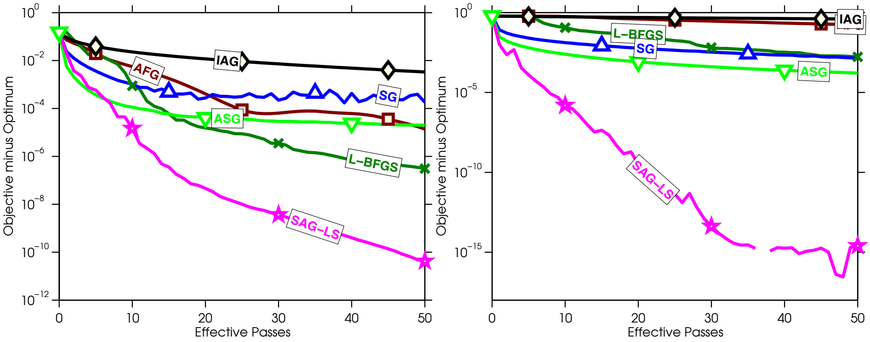
Comparing Deterministic and Stochatic Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



SAG Compared to FG and SG Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



Other Linearly-Convergent Stochastic Methods

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

Other Linearly-Convergent Stochastic Methods

- Newer stochastic algorithms are now available with linear rates:
  - Stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
  - Incremental surrogate optimization [Mairal, 2013].
  - Stochastic variance-reduced gradient (SVRG) [Johnson & Zhang, 2013, Konecny & Richtarik, 2013, Mahdavi et al., 2013, Zhang et al., 2013]
  - SAGA [Defazio et al., 2014]
- SVRG has a much lower memory requirement (later in talk).
- There are also non-smooth extensions (last part of talk).

SAG Implementation Issues

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

SAG Implementation Issues

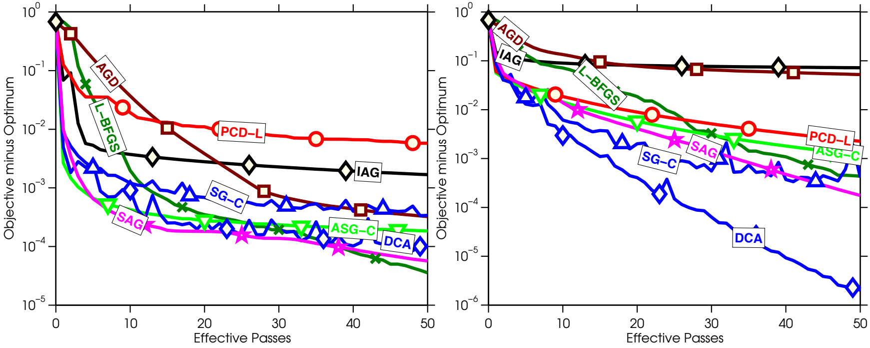
- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N}$.
- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
  - Termination criterion.
  - Acceleration [Lin et al., 2015].

SAG Implementation Issues

- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N}$.
- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
  - Termination criterion.
  - Acceleration [Lin et al., 2015].
  - Adaptive non-uniform sampling [Schmidt et al., 2013]:
    - Sample gradients that change quickly more often.

SAG with Adaptive Non-Uniform Sampling

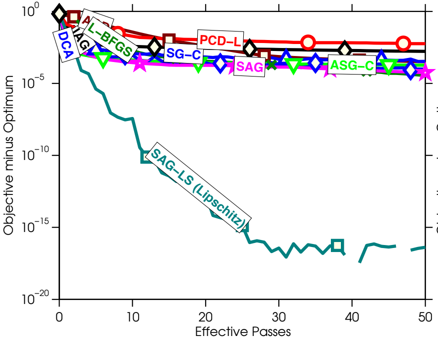
- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)

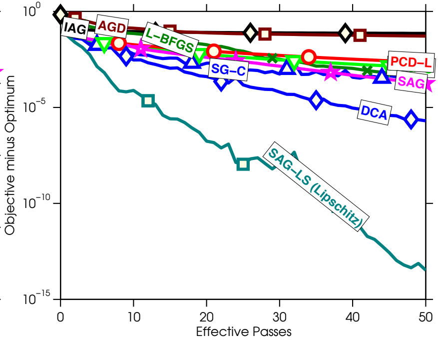


- Datasets where SAG had the worst relative performance.

SAG with Non-Uniform Sampling

- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)





- Lipschitz sampling helps a lot.

Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.

Minimizing Finite Sums: Dealing with the Memory

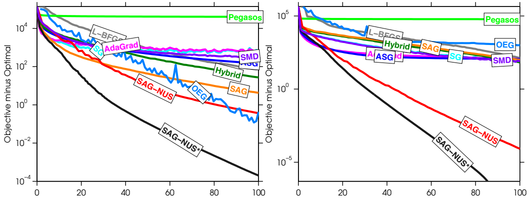
- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
    - Use mini-batches (only store gradient of the mini-batch).

Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches (only store gradient of the mini-batch).
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.

Minimizing Finite Sums: Dealing with the Memory

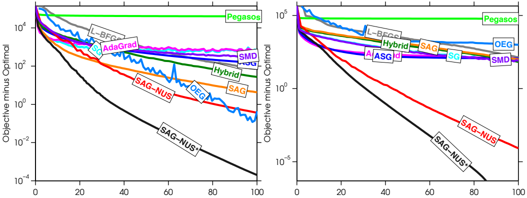
- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches (only store gradient of the mini-batch).
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.
    - For CRFs, only need to store marginals of parts.



(optical character and named-entity recognition tasks)

Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches (only store gradient of the mini-batch).
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.
    - For CRFs, only need to store marginals of parts.



(optical character and named-entity recognition tasks)

- If the above don't work, use SVRG...

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \dots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \dots$
  - $d_s = \frac{1}{N}\sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$
  - for $t = 1, 2, \dots m$
    - Randomly pick $i_t \in \{1, 2, \dots, N\}$
    - $x^t = x^{t-1} - \alpha_t(f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \dots, m\}$.

Stochastic Variance-Reduced Gradient

SVRG algorithm:

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

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

Outline

Motivation: Sparse Regularization

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Often, regularizer r is used to encourage sparsity pattern in x.

Motivation: Sparse Regularization

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Often, regularizer r is used to encourage sparsity pattern in x.
- For example, ℓ_1-regularized least squares,

$$\min_x \|Ax - b\|^2 + \lambda \|x\|_1$$

- Regularizes and encourages sparsity in x

Motivation: Sparse Regularization

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Often, regularizer r is used to encourage sparsity pattern in x.
- For example, ℓ_1-regularized least squares,

$$\min_x \|Ax - b\|^2 + \lambda \|x\|_1$$

- Regularizes and encourages sparsity in x
- The objective is non-differentiable when any $x_i = 0$.
- Subgradient methods are optimal (slow) black-box methods.

Motivation: Sparse Regularization

- Recall the regularized empirical risk minimization problem:

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

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Often, regularizer r is used to encourage sparsity pattern in x.
- For example, ℓ_1-regularized least squares,

$$\min_x \|Ax - b\|^2 + \lambda \|x\|_1$$

- Regularizes and encourages sparsity in x
- The objective is non-differentiable when any $x_i = 0$.
- Subgradient methods are optimal (slow) black-box methods.
- Faster methods for specific non-smooth problems?

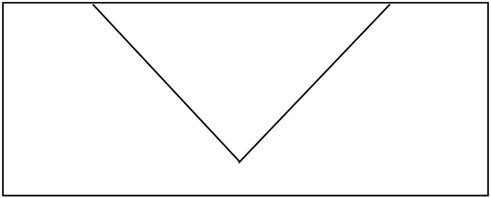
Smoothing Approximations of Non-Smooth Functions

- Smoothing: replace non-smooth f with smooth f_ϵ.
- Apply a fast method for smooth optimization.

Smoothing Approximations of Non-Smooth Functions

- Smoothing: replace non-smooth f with smooth f_ϵ.
- Apply a fast method for smooth optimization.
- Smooth approximation to the absolute value:

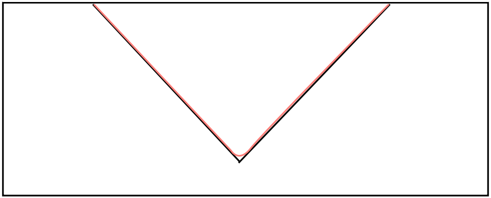
$$|x| \approx \sqrt{x^2 + \nu}.$$



Smoothing Approximations of Non-Smooth Functions

- Smoothing: replace non-smooth f with smooth f_ϵ.
- Apply a fast method for smooth optimization.
- Smooth approximation to the absolute value:

$$|x| \approx \sqrt{x^2 + \nu}.$$



Smoothing Approximations of Non-Smooth Functions

- Smoothing: replace non-smooth f with smooth f_ϵ.
- Apply a fast method for smooth optimization.
- Smooth approximation to the absolute value:

$$|x| \approx \sqrt{x^2 + \nu}.$$

- Smooth approximation to the max function:

$$\max\{a, b\} \approx \log(\exp(a) + \exp(b))$$

- Smooth approximation to the hinge loss:

$$\max\{0, x\} \approx \begin{cases} 0 & x \geq 1 \\ 1 - x^2 & t < x < 1 \\ (1-t)^2 + 2(1-t)(t-x) & x \leq t \end{cases}$$

Smoothing Approximations of Non-Smooth Functions

- Smoothing: replace non-smooth f with smooth f_ϵ.
- Apply a fast method for smooth optimization.
- Smooth approximation to the absolute value:

$$|x| \approx \sqrt{x^2 + \nu}.$$

- Smooth approximation to the max function:

$$\max\{a, b\} \approx \log(\exp(a) + \exp(b))$$

- Smooth approximation to the hinge loss:

$$\max\{0, x\} \approx \begin{cases} 0 & x \geq 1 \\ 1 - x^2 & t < x < 1 \\ (1-t)^2 + 2(1-t)(t-x) & x \leq t \end{cases}$$

- Generic smoothing strategy: strongly-convex regularization of convex conjugate.[Nesterov, 2005]

Discussion of Smoothing Approach

- Nesterov [2005] shows that:
  - Gradient method on smoothed problem has $O(1/\sqrt{t})$ subgradient rate.
  - Accelerated gradient method has faster $O(1/t)$ rate.

Discussion of Smoothing Approach

- Nesterov [2005] shows that:
  - Gradient method on smoothed problem has $O(1/\sqrt{t})$ subgradient rate.
  - Accelerated gradient method has faster $O(1/t)$ rate.
- In practice:
  - Slowly decrease level of smoothing (often difficult to tune).
  - Use faster algorithms like L-BFGS, SAG, or SVRG.

Discussion of Smoothing Approach

- Nesterov [2005] shows that:
  - Gradient method on smoothed problem has $O(1/\sqrt{t})$ subgradient rate.
  - Accelerated gradient method has faster $O(1/t)$ rate.
- In practice:
  - Slowly decrease level of smoothing (often difficult to tune).
  - Use faster algorithms like L-BFGS, SAG, or SVRG.
- You can get the $O(1/t)$ rate for $\min_x \max\{f_i(x)\}$ for f_i convex and smooth using *mirror-prox* method.[Nemirovski, 2004]
  - See also Chambolle & Pock [2010].

Converting to Constrained Optimization

- Re-write non-smooth problem as constrained problem.

Converting to Constrained Optimization

- Re-write non-smooth problem as constrained problem.
- The problem

$$\min_x f(x) + \lambda\|x\|_1,$$

is equivalent to the problem

$$\min_{x^+\geq 0, x^-\geq 0} \quad f(x^+ - x^-) + \lambda \sum_i (x_i^+ + x_i^-),$$

or the problems

$$\min_{-y\leq x\leq y} \quad f(x) + \lambda \sum_i y_i, \qquad \min_{\|x\|_1\leq\gamma} \quad f(x) + \lambda\gamma$$

Converting to Constrained Optimization

- Re-write non-smooth problem as constrained problem.
- The problem

$$\min_x f(x) + \lambda\|x\|_1,$$

is equivalent to the problem

$$\min_{x^+\geq 0, x^-\geq 0} \quad f(x^+ - x^-) + \lambda \sum_i (x_i^+ + x_i^-),$$

or the problems

$$\min_{-y\leq x\leq y} \quad f(x) + \lambda \sum_i y_i, \qquad \min_{\|x\|_1\leq\gamma} \quad f(x) + \lambda\gamma$$

- These are smooth objective with 'simple' constraints.

$$\min_{x\in\mathcal{C}} f(x).$$

Optimization with Simple Constraints

- Recall: gradient descent minimizes quadratic approximation:

$$x^{t+1} = \operatorname*{argmin}_y \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}.$$

Optimization with Simple Constraints

- Recall: gradient descent minimizes quadratic approximation:

$$x^{t+1} = \underset{y}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}.$$

- Consider minimizing subject to simple constraints:

$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}.$$

Optimization with Simple Constraints

- Recall: gradient descent minimizes quadratic approximation:

$$x^{t+1} = \underset{y}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}.$$

- Consider minimizing subject to simple constraints:

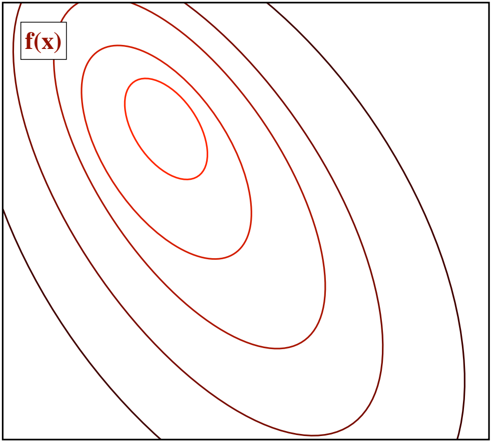
$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\}.$$

- Equivalent to projection of gradient descent:

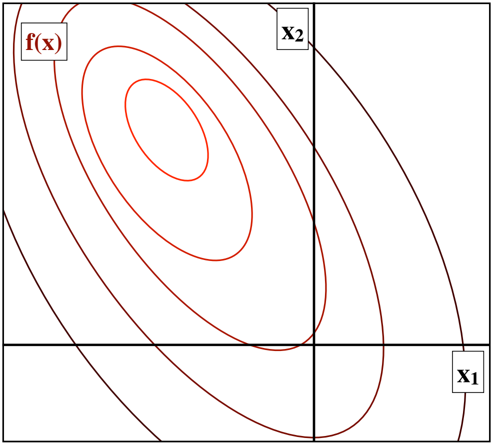
$$x_t^{GD} = x^t - \alpha_t \nabla f(x^t),$$

$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ \|y - x_t^{GD}\| \right\},$$

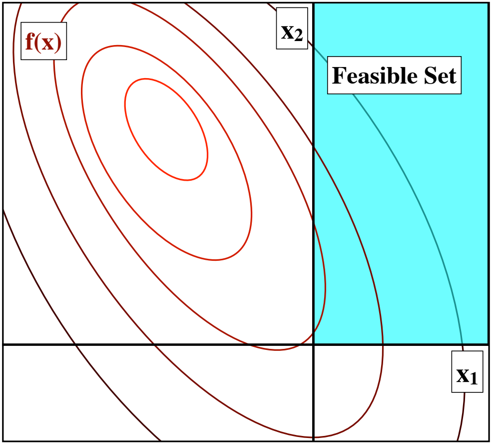
Gradient Projection



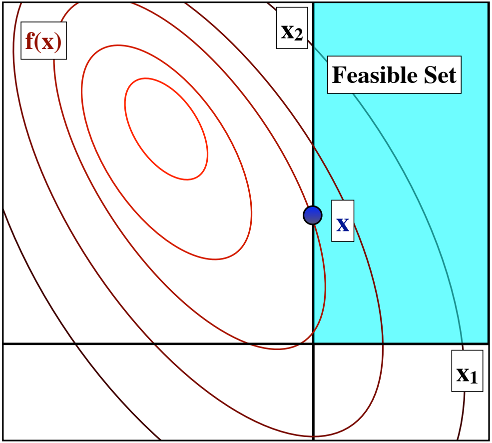
Gradient Projection



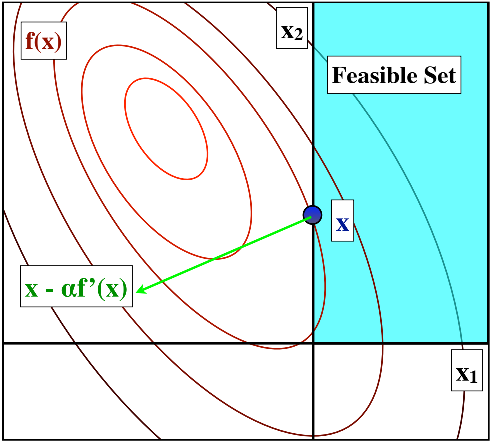
Gradient Projection



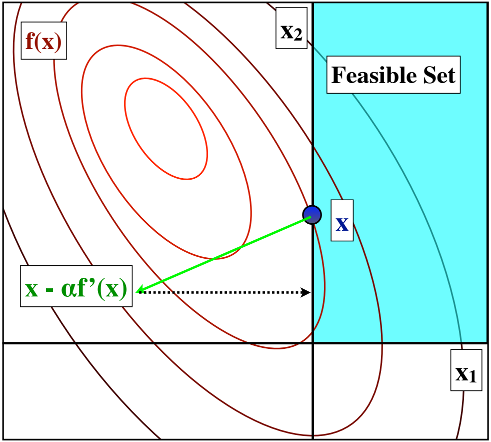
Gradient Projection



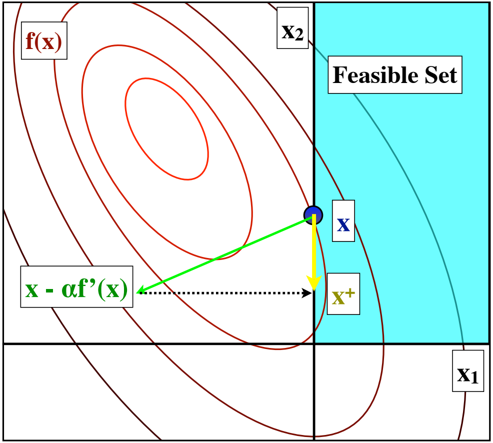
Gradient Projection



Gradient Projection



Gradient Projection



Discussion of Projected Gradient

- Projected gradient has same rate as gradient method!

Discussion of Projected Gradient

- Projected gradient has same rate as gradient method!
- Can do many of the same tricks (i.e. line-search, acceleration, Barzilai-Borwein, SAG, SVRG).

Discussion of Projected Gradient

- Projected gradient has same rate as gradient method!
- Can do many of the same tricks (i.e. line-search, acceleration, Barzilai-Borwein, SAG, SVRG).
- For projected Newton, you need to do an expensive projection under $\|\cdot\|_{H_t}$.
  - Two-metric projection methods allow Newton-like strategy for bound constraints.
  - Inexact Newton methods allow Newton-like like strategy for optimizing costly functions with simple constraints.

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.
- $\text{argmin}_{a^T y \geq b} \|y - x\| = \begin{cases} x & a^T x \geq b \\ x + (b - a^T x)a/\|a\|^2 & a^T x < b \end{cases}$

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.
- $\text{argmin}_{a^T y \geq b} \|y - x\| = \begin{cases} x & a^T x \geq b \\ x + (b - a^T x)a/\|a\|^2 & a^T x < b \end{cases}$
- $\text{argmin}_{\|y\| \leq \tau} \|y - x\| = \tau x/\|x\|$.

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.
- $\text{argmin}_{a^T y \geq b} \|y - x\| = \begin{cases} x & a^T x \geq b \\ x + (b - a^T x)a/\|a\|^2 & a^T x < b \end{cases}$
- $\text{argmin}_{\|y\| \leq \tau} \|y - x\| = \tau x/\|x\|$.
- Linear-time algorithm for ℓ_1-norm $\|y\|_1 \leq \tau$.

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.
- $\text{argmin}_{a^T y \geq b} \|y - x\| = \begin{cases} x & a^T x \geq b \\ x + (b - a^T x)a/\|a\|^2 & a^T x < b \end{cases}$
- $\text{argmin}_{\|y\| \leq \tau} \|y - x\| = \tau x/\|x\|$.
- Linear-time algorithm for ℓ_1-norm $\|y\|_1 \leq \tau$.
- Linear-time algorithm for probability simplex $y \geq 0, \sum y = 1$.

Projection Onto Simple Sets

Projections onto simple sets:

- $\text{argmin}_{y \geq 0} \|y - x\| = \max\{x, 0\}$
- $\text{argmin}_{l \leq y \leq u} \|y - x\| = \max\{l, \min\{x, u\}\}$
- $\text{argmin}_{a^T y = b} \|y - x\| = x + (b - a^T x)a/\|a\|^2$.
- $\text{argmin}_{a^T y \geq b} \|y - x\| = \begin{cases} x & a^T x \geq b \\ x + (b - a^T x)a/\|a\|^2 & a^T x < b \end{cases}$
- $\text{argmin}_{\|y\| \leq \tau} \|y - x\| = \tau x/\|x\|$.
- Linear-time algorithm for ℓ_1-norm $\|y\|_1 \leq \tau$.
- Linear-time algorithm for probability simplex $y \geq 0, \sum y = 1$.
- Intersection of simple sets: Dykstra's algorithm.

We can solve large instances of problems with these constraints.

Proximal-Gradient Method

- A generalization of projected-gradient is Proximal-gradient.
- The proximal-gradient method addresses problem of the form

$$\min_x f(x) + r(x),$$

where f is smooth but r is a general convex function.

Proximal-Gradient Method

- A generalization of projected-gradient is Proximal-gradient.
- The proximal-gradient method addresses problem of the form

$$\min_x f(x) + r(x),$$

where f is smooth but r is a general convex function.

- Applies proximity operator of r to gradient descent on f:

$$x_t^{GD} = x^t - \alpha_t \nabla f(x_t),$$

$$x^{t+1} = \operatorname*{argmin}_y \left\{ \frac{1}{2}\|y - x_t^{GD}\|^2 + \alpha r(y) \right\},$$

Proximal-Gradient Method

- A generalization of projected-gradient is Proximal-gradient.
- The proximal-gradient method addresses problem of the form

$$\min_x f(x) + r(x),$$

where f is smooth but r is a general convex function.

- Applies proximity operator of r to gradient descent on f:

$$x_t^{GD} = x^t - \alpha_t \nabla f(x_t),$$

$$x^{t+1} = \underset{y}{\operatorname{argmin}} \left\{ \frac{1}{2}\|y - x_t^{GD}\|^2 + \alpha r(y) \right\},$$

- Equivalent to using the approximation

$$x^{t+1} = \underset{y}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha}\|y - x^t\|^2 + r(y) \right\}.$$

- Convergence rates are still the same as for minimizing f.

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x \in \mathbb{R}^P}{\text{argmin}} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x \in \mathbb{R}^P}{\text{argmin}} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^{t+1} = \text{softThresh}_{\alpha\lambda}[x^t - \alpha \nabla f(x^t)].$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x \in \mathbb{R}^P}{\text{argmin}} \; r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^{t+1} = \text{softThresh}_{\alpha\lambda}[x^t - \alpha \nabla f(x^t)].$$

- Example with $\lambda = 1$:

Input Threshold Soft-Threshold

$$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x\in\mathbb{R}^P}{\text{argmin}}\ r(x) + \frac{1}{2}\|x-y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^{t+1} = \text{softThresh}_{\alpha\lambda}[x^t - \alpha\nabla f(x^t)].$$

- Example with $\lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[y] = \underset{x \in \mathbb{R}^P}{\text{argmin}}\; r(x) + \frac{1}{2}\|x - y\|^2.$$

- For L1-regularization, we obtain iterative soft-thresholding:

$$x^{t+1} = \text{softThresh}_{\alpha\lambda}[x^t - \alpha \nabla f(x^t)].$$

- Example with $\lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.2075 \\ 0 \\ 0.6302 \\ 0 \end{bmatrix}$

Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$

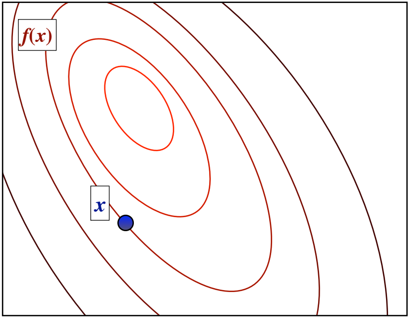
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$



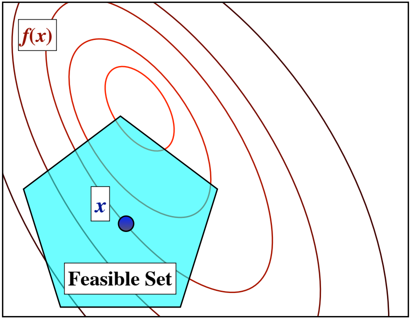
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$



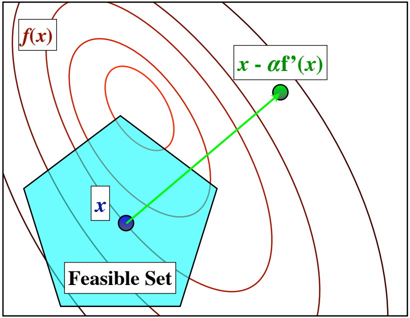
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$



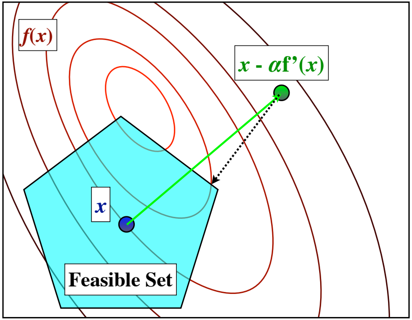
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$



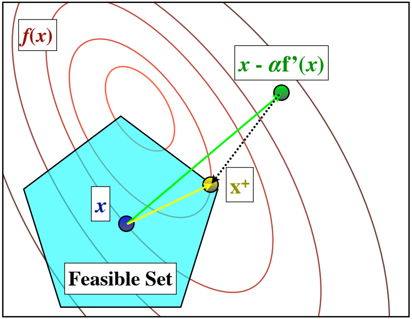
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

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

gives

$$x^{t+1} = \text{project}_{\mathcal{C}}[x^t - \alpha \nabla f(x^t)],$$



Exact Proximal-Gradient Methods

- For what problems can we apply these methods?

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
    1. L1-Regularization.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. Small number of linear constraint.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  6. A few other simple regularizers/constraints.

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  6. A few other simple regularizers/constraints.
- Can solve these non-smooth/constrained problems as fast as smooth/unconstrained problems!

Exact Proximal-Gradient Methods

- For what problems can we apply these methods?
- We can efficiently compute the proximity operator for:
  1. L1-Regularization.
  2. Group ℓ_1-Regularization.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  6. A few other simple regularizers/constraints.
- Can solve these non-smooth/constrained problems as fast as smooth/unconstrained problems!
- We can again do many of the same tricks (line-search, acceleration, Barzilai-Borwein, two-metric projection, inexact proximal operators, SAG, SVRG).

Alternating Direction Method of Multipliers

- Alernating direction method of multipliers (ADMM) solves:

$$\min_{Ax+By=c} f(x) + r(y).$$

- Alternate between prox-like operators with respect to f and r.

Alternating Direction Method of Multipliers

- Alernating direction method of multipliers (ADMM) solves:

$$\min_{Ax+By=c} f(x) + r(y).$$

- Alternate between prox-like operators with respect to f and r.
- Can introduce constraints to convert to this form:

$$\min_{x} f(Ax) + r(x) \quad \Leftrightarrow \quad \min_{x=Ay} f(x) + r(y),$$

Alternating Direction Method of Multipliers

- Alernating direction method of multipliers (ADMM) solves:

$$\min_{Ax+By=c} f(x) + r(y).$$

- Alternate between prox-like operators with respect to f and r.
- Can introduce constraints to convert to this form:

$$\min_x f(Ax) + r(x) \quad \Leftrightarrow \quad \min_{x=Ay} f(x) + r(y),$$

$$\min_x f(x) + r(Bx) \quad \Leftrightarrow \quad \min_{y=Bx} f(x) + r(y).$$

Alternating Direction Method of Multipliers

- Alernating direction method of multipliers (ADMM) solves:

$$\min_{Ax+By=c} f(x) + r(y).$$

- Alternate between prox-like operators with respect to f and r.
- Can introduce constraints to convert to this form:

$$\min_{x} f(Ax) + r(x) \quad \Leftrightarrow \quad \min_{x=Ay} f(x) + r(y),$$

$$\min_{x} f(x) + r(Bx) \quad \Leftrightarrow \quad \min_{y=Bx} f(x) + r(y).$$

- If prox can not be computed exactly: Linearized ADMM.

Frank-Wolfe Method

- In some cases the projected gradient step

$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\},$$

may be hard to compute (e.g., dual of max-margin Markov networks).

Frank-Wolfe Method

- In some cases the projected gradient step

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t}\|y - x^t\|^2 \right\},$$

may be hard to compute (e.g., dual of max-margin Markov networks).

- Frank-Wolfe method simply uses:

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) \right\},$$

requires compact $\mathcal{C}$, takes convex combination of x^t and x^{t+1}.

Frank-Wolfe Method

- In some cases the projected gradient step

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\},$$

  may be hard to compute (e.g., dual of max-margin Markov networks).

- Frank-Wolfe method simply uses:

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) \right\},$$

  requires compact $\mathcal{C}$, takes convex combination of x^t and x^{t+1}.

- Iterate can be written as convex combination of vertices of $\mathcal{C}$.
- $O(1/t)$ rate for smooth convex objectives, some linear convergence results for smooth and strongly-convex.[Jaggi, 2013]

Alternatives to Quadratic/Linear Surrogates

- Mirror descent uses the iterations[Beck & Teboulle, 2003]

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x) + \nabla f(x)^T (y - x^t) + \frac{1}{2\alpha_t} \mathcal{D}(x^t, y) \right\},$$

  where $\mathcal{D}$ is a Bregman-divergence:
  - $\mathcal{D} = \|x^t - y\|^2$ (gradient method).
  - $\mathcal{D} = \|x^t - y\|_H^2$ (Newton's method).
  - $\mathcal{D} = \sum_i x_i^t \log(\frac{x_i^t}{y_i}) - \sum_i (x_i^t - y_i)$ (exponentiated gradient).

Alternatives to Quadratic/Linear Surrogates

- Mirror descent uses the iterations[Beck & Teboulle, 2003]

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \left\{ f(x) + \nabla f(x)^T (y - x^t) + \frac{1}{2\alpha_t} \mathcal{D}(x^t, y) \right\},$$

  where $\mathcal{D}$ is a Bregman-divergence:
  - $\mathcal{D} = \|x^t - y\|^2$ (gradient method).
  - $\mathcal{D} = \|x^t - y\|_H^2$ (Newton's method).
  - $\mathcal{D} = \sum_i x_i^t \log(\frac{x_i^t}{y_i}) - \sum_i (x_i^t - y_i)$ (exponentiated gradient).
- Mairal [2013,2014] considers general surrogate optimization:

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \{g(y)\},$$

  where g upper bounds f, $g(x^t) = f(x^t)$, $\nabla g(x^t) = \nabla f(x^t)$, and $\nabla g - \nabla f$ is Lipschitz-continuous.
  - Get $O(1/k)$ and linear convergence rates depending on $g - f$.

Dual Methods

- Stronly-convex problems have smooth duals.
- Solve the dual instead of the primal.

Dual Methods

- Stronly-convex problems have smooth duals.
- Solve the dual instead of the primal.
- SVM non-smooth strongly-convex primal:

$$\min_{x} C \sum_{i=1}^{N} \max\{0, 1 - b_i a_i^T x\} + \frac{1}{2}\|x\|^2.$$

- SVM smooth dual:

$$\min_{0 \leq \alpha \leq C} \frac{1}{2}\alpha^T A A^T \alpha - \sum_{i=1}^{N} \alpha_i$$

- Smooth bound constrained problem:
  - Two-metric projection (efficient Newton-liked method).
  - Randomized coordinate descent (part 2 of this talk).

Summary

Summary:

- Part 1: Convex functions have special properties that allow us to efficiently minimize them.
- Part 2: Gradient-based methods allow elegant scaling with dimensionality of problem.
- Part 3: Stochastic-gradient methods allow scaling with number of training examples, at cost of slower convergence rate.
- Part 4: For finite datasets, SAG fixes convergence rate of stochastic gradient methods, and SVRG fixes memory problem of SAG.
- Part 5: These building blocks can be extended to solve a variety of constrained and non-smooth problems.