Fall 2016

IE 598 Big Data Optimization

Lecture Notes

Niao He
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

What's Inside

Module I: Introduction and Fundamentals

- ♣ Lecture 1: Introduction
- ♣ Lecture 2: Convex Sets
- ♣ Lecture 3-4: Convex Functions
- ♣ Lecture 5: Convex Optimization
- ♣ Lecture 6: Conic Programming
- ♣ Lecture 7: Introduction to Optimization

Module II: Smooth Convex Optimization

- ♦ Lecture 8: Gradient Descent
- ♦ Lecture 9: Gradient Descent and Its Acceleration
- ♦ Lecture 10: Projected Gradient Descent
- ♦ Lecture 11: Conditional Gradient Algorithm
- ♦ Lecture 12: Coordinate Descent Algorithm
- ♦ Lecture 13: Case Study on Logistic Regression

Module III: Nonsmooth Convex Optimization

- ♥ Lecture 14: Subgradient Methods
- ♥ Lecture 15: Mirror Descent
- ♥ Lecture 16: Smoothing Technique I
- ♥ Lecture 17: Smoothing Technique II
- ♥ Lecture 18: Mirror Prox Algorithm
- ♥ Lecture 19: Proximal Gradient Method and Its Acceleration
- ♥ Lecture 20: Splitting Algorithms and Case Study on LASSO Regression

Module IV: Stochastic and Online Optimization

- ♠ Lecture 21: Stochastic Optimization I
- ♠ Lecture 22: Stochastic Optimization II
- ♠ Lecture 23: Incremental Gradient Algorithms
- ♠ Lecture 24: Summary and Outlook

IE 598: Big Data Optimization **Fall 2016**

Lecture 2: Basic Convex Analysis – August 25

Lecturer: Niao He *Scribes: Niao He*

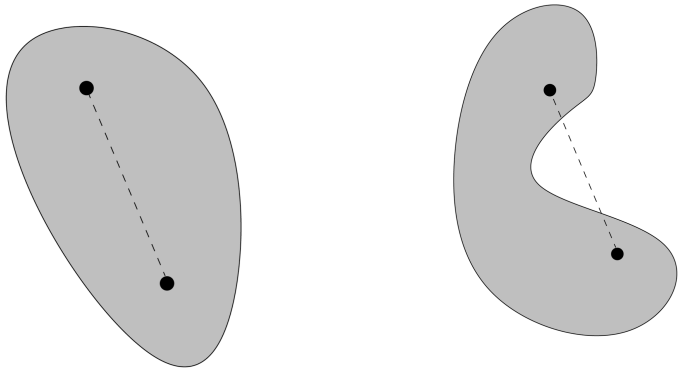
Overview In this lecture, we will introduce the concept of convex set, examples, calculus of convexity, and some geometry results.

2.1 Convex Sets

2.1.1 Definitions

Definition 2.1 (Convex set) *A set $X \subseteq \mathbf{R}^n$ is convex if $\forall x, y \in X, \lambda x + (1-\lambda)y \in X$ for any $\lambda \in [0,1]$.*

In another word, the line segment that connects any two elements lies entirely in the set.



(a) convex set (b) non-convex set

Figure 2.1: Examples of convex sets

Given any elements $x_1, \ldots, x_k$, the combination $\lambda_1 x_1 + \lambda_2 x_1 + \ldots + \lambda_k x_k$ is called

- **Covex**: if $\lambda_i \geq 0, i = 1, \ldots, k$ and $\lambda_1 + \lambda_2 + \ldots + \lambda_k = 1$;
- **Conic**: if $\lambda_i \geq 0, i = 1, \ldots, k$;
- **Linear**: if $\lambda_i \in \mathbf{R}, i = 1, \ldots, k$.

A set is *convex* if all convex combinations of its elements are in the set; a set is a *cone* if all conic combinations of its elements are in the set; a set is a *linear subspace* if all linear combinations of its elements are in the set. Clearly, a linear subspace is always a cone; a cone is always a convex set.

Definition 2.2 (Convex hull) *A convex hull of a set $X \subseteq \mathbf{R}^n$ is the set of all convex combination of its elements, denoted as*

$$conv(X) = \left\{ \sum_{i=1}^{k} \lambda_i x_i : k \in \mathbf{N}, \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1, x_i \in X, \forall i = 1, \ldots, k \right\}.$$

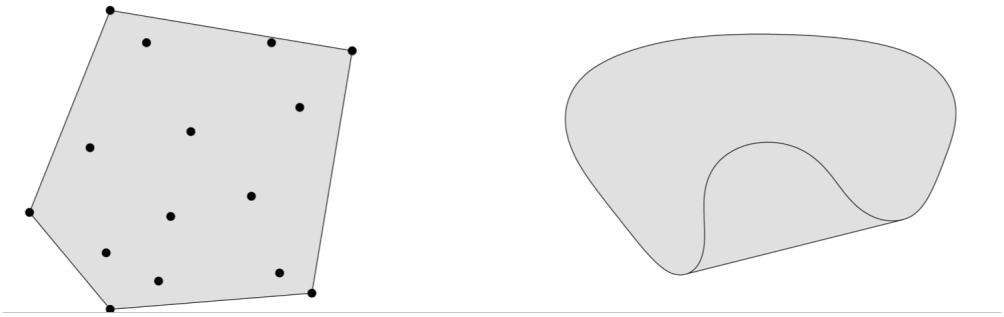


Figure 2.2: Examples of convex hulls

Remark. It follows immediately that

1. A convex hull is always convex.
2. If X is convex, then $conv(X) = X$.
3. For any set X, $conv(X)$ is the smallest convex set that contains X.

2.1.2 Examples of Convex Sets

Example 1. Some simple convex sets:

- *Hyperplane*: $\{x \in \mathbf{R}^n : a^T x = b\}$
- *Halfspace*: $\{x \in \mathbf{R}^n : a^T x \leq b\}$
- *Affine space*: $\{x \in \mathbf{R}^n : Ax = b\}$
- *Polyhedron*: $\{x \in \mathbf{R}^n : Ax \leq b\}$
- *Simplex*: $\{x \in \mathbf{R}^n : x \geq 0, \sum_{i=1}^{n} x_i = 1\} = conv(e_1, \ldots, e_n)$.

Example 2. *Norm balls*:

$$B_{\|\cdot\|} := \{x \in \mathbf{R}^n : \|x - a\| \leq r\}$$

for a given norm $\|\cdot\|$ on $\mathbf{R}^n$.

[Note that $\|\cdot\|$ is a norm if

1. (positivity): $\|x\| \geq 0$ and $\|x\| + 0$ if and only if $x = 0$;

2. (homogeneity): $\|\lambda x\| = |\lambda| \cdot \|x\|, \forall \lambda \in \mathbf{R}$;
3. (triangle inequality): $\|x + y\| \leq \|x\| + \|y\|, \forall x, y$

A typical example is the ℓ_p-*norm* ($1 \leq p \leq \infty$):

$$\|x\|_p = \left(\sum_{i=1}^{n} x_i^p \right)^{1/p}.$$

- when $p = 2$, $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ is known as the *Euclidean norm*;
- when $p = 1$, $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ is known as the ℓ_1-*norm*;
- when $p = \infty$, $\|x\|_\infty = \max\{|x_i| : i = 1, \ldots, n\}$ is known as the *max norm*.]

It follows immediately from the homogeneity and triangle inequality conditions that a norm ball is always convex. If $x, y \in B_{\|\cdot\|}$ and $\lambda \in [0, 1]$, then

$$\|[\lambda x + (1 - \lambda)y] - a\| \leq \lambda \|x - a\| + (1 - \lambda)\|y - a\| \leq \lambda r + (1 - \lambda)r = r.$$

Hence, $\lambda x + (1 - \lambda)y \in B_{\|\cdot\|}$.

Example 3. *Ellipsoid*:

$$\{x \in \mathbf{R}^n : (x - a)^T Q (x - a) \leq r^2\}$$

where $Q \succ 0$ and is symmetric.

Proof: Define $\|x\|_Q := \|Q^{1/2} x\|_2$, one can show that $\|x\|_Q$ is a valid norm. Since $(x-a)^T Q (x-a) = \|x-a\|_Q^2$, the ellipsoid can be considered as a special norm ball; hence it is convex. ■

Example 4. *ϵ-neighborhood of convex set*, i.e.

$$X^\epsilon := \{x \in \mathbf{R}^n : dist_{\|\cdot\|}(x, X) := \inf_{y \in X} \|x - y\| \leq \epsilon\}.$$

Proof:

$$\begin{aligned} x, y \in X^\epsilon &\Rightarrow \forall \epsilon' > \epsilon, \exists u, v \in X, \text{s. t. } \|x - u\| \leq \epsilon, \|y - v\| \leq \epsilon \\ &\Rightarrow \forall \lambda \in [0, 1], \exists \lambda u + (1 - \lambda)v \in X, \text{s. t. } \|[\lambda x + (1 - \lambda)y] - [\lambda u + (1 - \lambda)v]\| \leq \epsilon' \\ &\Rightarrow \forall \lambda \in [0, 1], \lambda x + (1 - \lambda)y \in X^\epsilon \end{aligned}$$

■

Example 5. *Convex cones*:

X is a convex cone if and only if $\forall x, y \in X, \lambda_1, \lambda_2 \geq 0, \lambda_1 x + \lambda_2 y \in X$.

- *Nonnegative orthant*: $\mathbf{R}^n_+ = \{x \in \mathbf{R}^n : x \geq 0\}$
- *Lorentz cone*: $L^n = \left\{ x \in \mathbf{R}^n : x_n \geq \sqrt{x_1^2 + \ldots + x_{n-1}^2} \right\}$

– *PSD cone*: $S_+^n = \{X \in \mathbf{S}^{n\times n} : X \succeq 0\}$

– *Polyhedral cone*: $P = \{x \in \mathbf{R}^n : Ax \leq 0\}$

You can verify that all of these are convex cones. Convex cones form an extremely important class of convex sets and enjoy many properties "parallel" to those of general convex sets.

2.1.3 Calculus of Convex Sets

The following operators preserve the convexity of sets, which can be easily verified based on the definition.

1. **Intersection**: If $X_\alpha, \alpha \in \mathcal{A}$ are convex sets, then

   $$\cap_{\alpha\in\mathcal{A}} X_\alpha$$

   is also a convex set.

2. **Direct product**: If $X_i \subseteq \mathbf{R}^{n_i}, i = 1, \ldots, k$ are convex sets, then

   $$X_1 \times \cdots \times X_k := \{(x_1, \ldots, x_k) : x_i \in X_i, i = 1, \ldots, k\}$$

   is also a convex set.

3. **Weighted summation**: If $X_i \subseteq \mathbf{R}^n, i = 1, \ldots, k$ are convex sets, then

   $$\lambda_1 X_1 + \cdots + \lambda_k X_k := \{\lambda_1 x_1 + \ldots + \lambda_k x_k : x_i \in X_i, i = 1, \ldots, k\}$$

   is also a convex set.

4. **Affine image**: If $X \subseteq \mathbf{R}^n$ is a convex set and $\mathcal{A}(x) : x \mapsto Ax + b$ is an affine mapping from $\mathbf{R}^n$ to $\mathbf{R}^k$, then

   $$\mathcal{A}(X) := \{Ax + b : x \in X\}$$

   is also a convex set.

5. **Inverse affine image**: If $X \subseteq \mathbf{R}^n$ is a convex set and $\mathcal{A}(y) : y \mapsto Ay + b$ is an affine mapping from $\mathbf{R}^k$ to $\mathbf{R}^n$, then

   $$\mathcal{A}^{-1}(X) := \{y : Ay + b \in X\}$$

   is also a convex set.

Example 7. *Linear matrix inequalities*:

$$C := \{x \in \mathbf{R}^k : x_1 A_1 + \ldots + x_k A_k \preceq B\} = \{x \in \mathbf{R}^k : B - \sum\nolimits_{i=1}^k x_i A_i \succeq 0\}$$

where $A_1, \ldots, A_k, B \in \mathbf{S}^n$ are symmetric matrices. *Proof:* This is because C can be considered as the inverse affine image of the convex set $\mathbf{S}^n$ under the affine mapping $\mathcal{A} : x \mapsto B - \sum_{i=1}^k x_i A_i$, i.e. $C = \mathcal{A}^{-1}(\mathbf{S}^n)$. ■

2.1.4 Geometry of Convex Sets

Convex sets are special because of their nice geometric properties. We cover some useful results below but without providing the proofs. Details can be found in [BV04,BN01].

Proposition 2.3 *If $X \neq \emptyset$, then*

1. *Both $int(X)$ and $cl(X)$ are convex;*
2. *If $int(X) \neq \emptyset$, then $int(X)$ is dense in $cl(X)$.*

Note that in general, for any set X, $int(X) \subseteq X \subseteq cl(X)$, but $int(X)$ and $cl(X)$ can differ dramatically. For instance, let X be the set of all irrational numbers in $(0, 1)$, then $int(X) = \emptyset$, $cl(X) = [0, 1]$. The proposition implies that a convex set is perfectly well characterized by its closure or interior if nonempty.

Theorem 2.4 (Carathedory) *Let $X \subseteq \mathbf{R}^n$ be any nonempty set. Then every point in $conv(X)$ is a convex combination of at most $dim(X) + 1$ points from X.*

Theorem 2.5 (Seperation) *Two nonempty convex sets X_1, X_2 can be separated by a hyperlane if $int(X_1) \cap \int(X_2) \neq \emptyset$, that is,*

$$\exists a, b, s.\ t.\ X_1 \subseteq \{x : a^T x \leq b\} \text{ and } X_2 \subseteq \{x : a^T x \geq b\}.$$

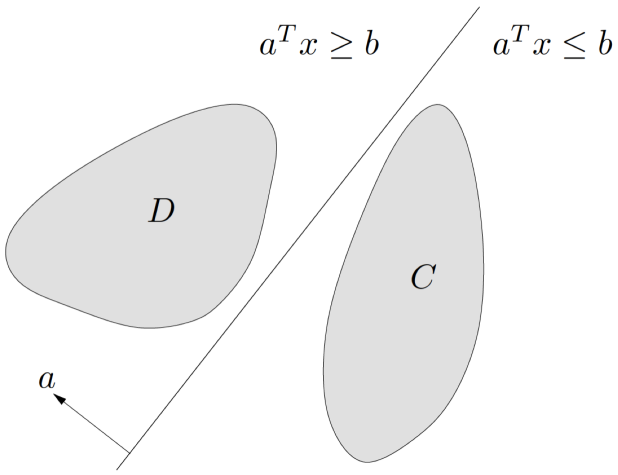


Figure 2.3: Separating hyperplane

Similarly, the supporting hyperplane theorem states that any boundary point of a convex set has a supporting hyperplane passing through it, i.e. if $\bar{x} \in cl(X)/int(X)$, then

$$\exists a, b, \text{s. t. } a^T \bar{x} = b \text{ and } a^T x \geq b, \forall x \in X.$$

References

[BV04] Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[BN01] Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, (Vol. 2). SIAM.

IE 598: Big Data Optimization **Fall 2016**

Lecture 3–4: Convex Functions – Aug 30 & Sep 1

Lecturer: Niao He *Scriber: Niao He*

Overview In these two lectures, we will introduce the concept of convex functions, and provide several ways to characterize convex functions, discuss some calculus that can be used to detect convexity of functions and compute the subgradients of convex function.

3.1 Definitions

Definition 3.1 (Convex function) *A function $f(x) : \mathbf{R}^n \to \mathbf{R}$ is convex if*

(i) $dom(f) \subseteq \mathbf{R}^n$ *s a convex set;*

(ii) $\forall x, y \in dom(f)$ *and* $\lambda \in [0, 1]$, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$.

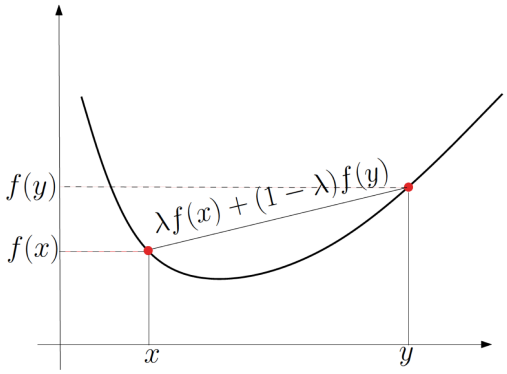


Figure 3.1: Example of convex function

A function is called strictly convex if (ii) holds with strict sign, i.e. $f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y)$.

A function is called α-strictly convex if $f(x) - \frac{\alpha}{2}\|x\|_2^2$ is convex.

A function is called concave if $-f(x)$ is convex.

For example, a linear function is both convex and concave. Any norm is convex.

Remark 1 (Extended value function). Conventionally, we can think of f as an extended value function from $\mathbf{R}^n$ to $\mathbf{R} \cup \{+\infty\}$ by setting $f(x) = +\infty$ if $x \notin dom(f)$, the condition (ii) is equivalent as

$$\forall x, y, \forall \lambda \in [0, 1], f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y).$$

Remark 2. (Slope inequality) What does convexity really mean? Let $z = \lambda x + (1-\lambda)y$, observe that $\|y-x\| : \|y-z\| : \|z-x\| = 1 : \lambda : (1-\lambda)$. Therefore

$$\begin{aligned} & f(z) \le \lambda f(x) + (1-\lambda) f(y) \\ \iff \quad & \frac{f(z)-f(x)}{1-\lambda} \le f(y) - f(x) \le \frac{f(y)-f(z)}{\lambda} \\ \iff \quad & \frac{f(z)-f(x)}{\|z-x\|} \le \frac{f(y)-f(x)}{\|y-x\|} \le \frac{f(y)-f(z)}{\|y-z\|} \end{aligned}$$

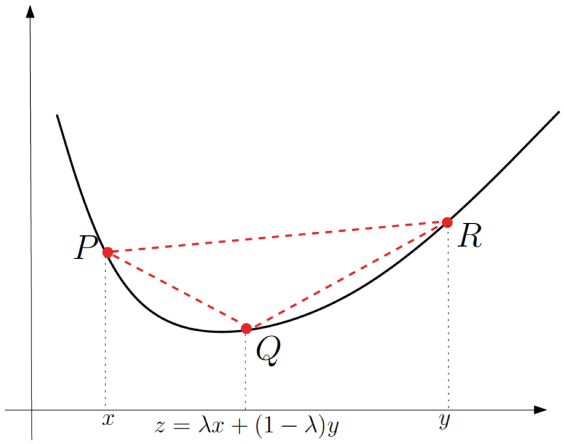


Figure 3.2: Slope PQ $\le$ Slope PR $\le$ Slope QR

3.2 Several Characterizations of Convex Functions

1. **Epigraph characterization**

   Proposition 3.2 *f is convex if and only if its epigraph*

   $$epi(f) := \{(x,t) \in \mathbf{R}^{n+1} : f(x) \le t\}$$

   is a convex set.

   Proof: This can be verified by using the definition of convex function and convex set.

   - ($\Longrightarrow$) Suppose $(x,t_1), (y,t_2) \in epi(f)$, then $f(x) \le t_1, f(y) \le t_2$. For any $\lambda \in [0,1]$, by convexity of f, $f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y) \le \lambda t_1 + (1-\lambda) t_2$. Thi implies that $\lambda \cdot (x,t_1) + (1-\lambda)\cdot(y,t_2) \in epi(f)$. Hence, $epi(f)$ is a convex set.
   - ($\Longleftarrow$) Let $x, y \in \mathbf{R}^n$, since $(x, f(x))$ and $(y, f(y))$ lie in $epi(f)$, by convexity of epigraph set, we have for any $\lambda \in [0,1]$, $(\lambda x + (1-\lambda)y, \lambda(x) + (1-\lambda) f(y)) \in epi(f)$. By definition, $f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y)$. Hence, function f is convex.

   ■

2. **Level set characterization**

   Proposition 3.3 *If f is convex, then the level set for any $t \in \mathbf{R}$*

   $$C_t(f) = \{x \in dom(f) : f(x) \leq t\}$$

   is a convex set.

   For example, the unit norm ball $\{x : \|x\| \leq 1\}$ is a convex set since $\|\cdot\|$ is convex.

   Remark. The reverse is not true. A function with convex level set is not always convex. In fact, it is known as a quasi-convex function.

3. **One-dimensional characterization**

   Proposition 3.4 *f is convex if and only if its restriction on any line, i.e. function*

   $$\phi(t) := f(x + th)$$

   is convex on the axis for any x and h.

   Remark. Convexity is a one-dimensional property. In order to detect the convexity of a function, it all boils down to check the convexity of a one-dimensional function on the axis. From basic calculus, we already know that

   $$\begin{aligned}
   & \phi(t) \text{ is convex on } (a,b) \\
   \iff & \frac{\phi(s)-\phi(t_1)}{s-t_1} \leq \frac{\phi(t_2)-\phi(t_1)}{t_2-t_1} \leq \frac{\phi(t_2)-\phi(s)}{t_2-s}, \forall a < t_1 < s < t_2 < b && \text{(due to slope inequality)} \\
   \iff & \phi'(t_1) \leq \phi'(t_2), \forall a < t_1 < t_2 < b && \text{(if } \phi \text{ is differentiable)} \\
   \iff & \phi''(t) > 0, \forall a < t < b && \text{(if } \phi \text{ is twice-differentiable)}
   \end{aligned}$$

   Hence, if f is differentiable or twice-differentiable, we can characterize it by based on its first-order or second-order.

4. **First-order characterization for differentiable convex functions**

   Proposition 3.5 *Assume f is differentiable, then f is convex if and only if $dom(f)$ is convex and for any x, y,*

   $$f(x) \geq f(y) + \nabla f(y)^T (x - y). \qquad (\star)$$

   Proof:

   - ($\Longrightarrow$) If f is convex, letting $z = (1-\epsilon)y + \epsilon x = y + \epsilon(x-y)$ with $\epsilon \in (0,1)$, from the slop inequality, we have
   $$\frac{f(x)-f(y)}{\|x-y\|} \geq \frac{f(z)-f(y)}{\|z-y\|} = \frac{f(y+\epsilon(x-y))-f(y)}{\epsilon\|x-y\|}.$$
   Hence, letting $\epsilon \to 0+$, we have
   $$f(x) - f(y) \geq \lim_{\epsilon \to 0+} \frac{f(y+\epsilon(x-y))-f(y)}{\epsilon} = \nabla f(y)^T (x-y).$$
   Therefore, $f(y) \geq f(x) + \nabla f(x)^T (y-x)$.

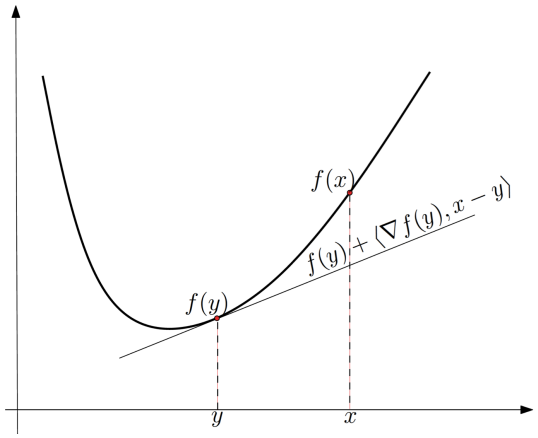


Figure 3.3: First-order condition

- ($\Longleftarrow$) If ($\star$) holds, letting $z = \lambda x + (1-\lambda)y$ for any $\lambda \in [0,1]$, we have
$$f(x) \geq f(z) + \nabla f(z)^T (x - z)$$
$$f(y) \geq f(z) + \nabla f(z)^T (y - z)$$
Adding the two inequalities with scalings λ and $(1-\lambda)$, it follows that
$$\lambda f(x) + (1-\lambda) f(y) \geq f(z) = f(\lambda x + (1-\lambda)y).$$
Hence, f is convex.

■

5. **Second-order characterization for twice-differentiable convex functions**

Proposition 3.6 *Assume f is twice-differentiable, then f is convex if and only if $dom(f)$ is convex and for any $x \in dom(f)$,*
$$\nabla^2 f(x) \succeq 0. \qquad (\star\star)$$

Proof:

- ($\Longrightarrow$) If f is convex, then for any x, h, $\phi(t) = f(x+th)$ is convex on the axis. Hence, $\phi''(t) \geq 0, \forall t$. Particularly,
$$\phi''(0) = h^T \nabla^2 f(x) h \geq 0.$$
This implies that $\nabla^2 f(x) \succeq 0$.
- ($\Longleftarrow$) It suffices to show that every one-dimensional function $\phi(t) := f(x + t(y-x))$ is convex for any $x, y \in dom(f)$. The latter is indeed true because $\phi''(t) = (y-x)^T \nabla^2 f(x+t(y-x))(y-x) \geq 0$ due to ($\star\star$).

■

6. **Subgradient characterization for non-differentiable convex functions**

Proposition 3.7 *f is convex if and only if $\forall x \in int(dom(f))$, there exists g, such that*
$$f(x) \geq f(y) + g^T (x - y)$$
i.e. the subdifferential set is non-empty.

To be discussed in Section 3.5.

3.3 Calculus of Convex Functions

The following operators preserve the convexity of functions, which can be easily verified based on the definition.

1. **Taking conic combination**: If $f_\alpha(x), \alpha \in \mathcal{A}$ are convex functions and $\{\lambda_\alpha\}_{\alpha\in\mathcal{A}} \geq 0$, then

$$\sum_{\alpha\in\mathcal{A}} \lambda_\alpha f_\alpha(x)$$

   is also a convex function.

2. **Taking affine composition** If $f(x)$ is convex on $\mathbf{R}^n$, and $\mathcal{A}(y) : y \mapsto Ay + b$ is an affine mapping from $\mathbf{R}^k$ to $\mathbf{R}^n$, then

$$g(y) := f(Ay + b)$$

   is convex on $\mathbf{R}^k$.

   The proofs are straightforward and hence omitted.

3. **Taking superposition**:

   - If f is a convex function on $\mathbf{R}^n$ and $F(\cdot)$ is a convex and non-decreasing function on $\mathbf{R}$, then $g(x) = F(f(x))$ is convex.
   - More generally, if $f_i(x), i = 1, \ldots, m$ are convex on $\mathbf{R}^n$ and $F(y_1, \ldots, y_m)$ is convex and non-decreasing (component-wise) on $\mathbf{R}^m$, then

$$g(x) = F(f_1(x), \ldots, f_m(x))$$

     is convex.

   Proof: By convexity of f_i, we have

$$f_i(\lambda x + (1-\lambda)y) \leq \lambda f_i(x) + (1-\lambda) f_i(y), \forall i, \forall \lambda \in [0,1].$$

   Hence, we have for any $\lambda \in [0,1]$,

$$\begin{aligned} g(\lambda x + (1-\lambda)y) &= F(f_1(\lambda x + (1-\lambda)y), \ldots, f_m(\lambda x + (1-\lambda)y)) \\ &\leq F(\lambda f_1(x) + (1-\lambda) f_1(y), \ldots, \lambda f_m(x) + (1-\lambda) f_m(y)) && (\text{ by monotonicity of } F) \\ &\leq \lambda F(f_1(x), \ldots, f_m(x)) + (1-\lambda) F(f_1(x), \ldots, f_m(x)) && (\text{ by convexity of } F) \\ &= \lambda g(x) + (1-\lambda) g(y) && (\text{ by definition of } g) \end{aligned}$$

   ■

4. **Taking supremum**: If $f_\alpha(x), \alpha \in \mathcal{A}$ are convex, then

$$\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$$

   is convex.

   Note that when $\mathcal{A}$ is finite, this can be considered as a special superposition with $F(y_1, \ldots, y_m) = \max(y_1, \ldots, y_m)$, which can be easily shown to be monotonic and convex.

Proof: We show that

$$\begin{aligned} epi(\sup_{\alpha\in\mathcal{A}} f_\alpha) &= \{(x,t) : \sup_{\alpha\in\mathcal{A}} f_\alpha(x) \le t\} \\ &= \{(x,t) : f_\alpha(x) \le t, \forall \alpha \in \mathcal{A}\} \\ &= \cap_{\alpha\in\mathcal{A}}\{(x,t) : f_\alpha(x) \le t\} \\ &= \cap_{\alpha\in\mathcal{A}} epi(f_\alpha). \end{aligned}$$

Since f_α is convex, $epi(f_\alpha)$ is therefore a convex set for any $\alpha \in \mathcal{A}$. Their intersection remains convex, i.e. $epi(\sup_{\alpha\in\mathcal{A}} f_\alpha)$ is a convex set, i.e. $\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ is convex. ■

5. **Partial minimization**: If $f(x,y)$ is convex in $(x,y) \in \mathbf{R}^n$ and C is a convex set, then

$$g(x) = \inf_{y\in C} f(x,y)$$

is convex.

Proof: Given any x_1, x_2, by definition, for any $\epsilon > 0$,

$$\begin{aligned} \exists y_1 : f(x_1,y_1) \le g(x_1) + \epsilon/2 \\ \exists y_2 : f(x_2,y_2) \le g(x_2) + \epsilon/2 \end{aligned}$$

For any $\lambda \in [0,1]$, adding the two equations, we have

$$\lambda f(x_1,y_1) + (1-\lambda) f(x_2,y_2) \le \lambda g(x_1) + (1-\lambda) g(x_2) + \epsilon.$$

Invoking the convexity of $f(x,y)$, this implies

$$f(\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2) \le \lambda g(x_1) + (1-\lambda) g(x_2) + \epsilon.$$

Hence for any $\epsilon > 0$, $g(\lambda x_1 + (1-\lambda)x_2) \le \lambda g(x_1) + (1-\lambda) g(x_2) + \epsilon$. Letting $\epsilon \to 0$ leads to the convexity of g. ■

6. **Perspective function**: If $f(x)$ is convex , then the *perspective* of f

$$g(x,t) := tf(x/t)$$

is convex on its domain $dom(g) = \{(x,t) : x/t \in dom(f), t > 0\}$.

Proof: Observe that

$$(x,t,\tau) \in epi(g) \Longleftrightarrow tf(x/t) < \tau \Longleftrightarrow f(x/t) \le \tau/t \Longleftrightarrow (x/t, \tau/t) \in epi(f)$$

Define the perspective function $P : \mathbf{R}^n \times \mathbf{R}_{++} \times \mathbf{R} :\to \mathbf{R}^n \times \mathbf{R}, (x,t,\tau) \mapsto (x/t, \tau/t)$, then

$$epi(g) = P^{-1}(epi(f)).$$

Since f is convex, $epi(f)$ is a convex set. To show g is convex, it suffices to show that the inverse image of a convex set under the perspective function is convex.

Claim: If U is a convex set, then

$$P^{-1}(U) = \{(u,t) : u/t \in U, t > 0\}$$

is a convex set.
This is because if $(u,t) \in P^{-1}(U)$ and $(v,t) \in P^{-1}(U)$, for any $\lambda \in [0,1]$,

$$\frac{\lambda u + (1-\lambda)v}{\lambda t + (1-\lambda)s} = \mu \cdot \frac{u}{t} + (1-\mu) \cdot \frac{v}{s} \in U$$

where $\mu = \frac{\lambda t}{\lambda t + (1-\lambda)s} \in [0,1]$. Hence, $\lambda \cdot (u,t) + (1-\lambda) \cdot (v,s) \in P^{-1}(U)$. ■

3.4 Examples of Convex Functions

Example 1. *Simple univariate functions:*

- $x^2, x^4, \ldots$
- e^{ax} for any a
- $-\log(x)$
- $x\log(x)$

Example 2. *Multi-variate functions*:

- $\|\cdot\|$
- $\frac{1}{2}x^TQx + b^Tx + c$, when $Q \succeq 0$
- $\|Ax - b\|_2^2$
- $\max(a_1^Tx + b_1, \ldots, a_k^Tx - b_k)$
- relative entropy function $g(x,t) : \mathbf{R}^2_{++} \to \mathbf{R}, (x,t) \mapsto t\log(t) - t\log(x)$
- $\log(\sum_{i=1}^k e^{a_i^Tx+b_i})$

  Proof: It suffices to show that $f(x) = \log(\sum_{i=1}^n e^{x_i})$ is convex. Observe that any h, we have

  $$h^T\nabla^2 f(x)h = \frac{\sum_i e^{x_i}h_i^2}{\sum_i e^{x_i}} - \frac{(\sum_i e^{x_i}h_i)^2}{(\sum_i e^{x_i})^2}.$$

  Let $p_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$, we have

  $$h^T\nabla^2 f(x)h = \sum_i p_ih_i^2 - (\sum_i p_ih_i)^2 \geq \sum_i p_ih_i^2 - \sum_i(\sqrt{p_i})^2\sum_i(\sqrt{p_i}h_i)^2 = \sum_i p_ih_i^2 - 1\cdot\sum_i p_ih_i^2 = 0.$$

  The first inequality is due to Cauchy-Schwarz inequality. Hence, $\nabla^2 f(x) \succeq 0$. ■

- $-\log(det(X))$

  Proof: Let $f(X) = -\log(det(X))$, the domain $dom(f) = S^n_{++}$. Let $X, H \succ 0$, it is sufficient to show that $g(t) = f(X + tH)$ is convex on $dom(g) = \{t : X + tH \succ 0\}$. Since

  $$g(t) = -\log(det(X+tH)) = -\log(det(X^{1/2}(I+tX^{-1/2}HX^{-1/2})X^{1/2})) = -\sum_i \log(1+t\lambda_i) - \log(det(X))$$

  where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $X^{-1/2}HX^{-1/2}$. Note that for each i, $-\log(1 + t\lambda_i)$ is convex in t, so $g(t)$ is also convex. ■

Example 3. Some distances:

- maximum distance to any set C: $d(x, C) := \max_{y\in C}\|x - y\|$
- minimum distance to a convex set C: $d(x, C) : \min_{y\in C}\|x - y\|$

Example 4. *Indicator and support functions*:

- indicator function of a convex set C: $I_C(x) := \begin{cases} 0, & x \in C \\ \infty, & x \notin C \end{cases}$
- support function of any set C (convex or not): $I_C^*(x) = \sup_{y \in C} x^T y$

3.5 Subgradients of Convex Functions

Definition 3.8 (Subgradient) *Let f be a convex function and $x \in dom(f)$, any vector g satisfying*

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

is called a subgradient of f at x.

The set of all subgradients of f at x is called the subdifferential, denoted as $\partial f(x)$.

Example 1. If f is differentiable at $x \in dom(f)$, then $\nabla f(x)$ is the unique element of $\partial f(x)$.
Proof: Let $g \in \partial f(x)$, by definition, $\frac{f(x+td)-f(x)}{t} \geq g^T d, , \forall d$. Let $t \to 0$, we have $\nabla f(x)^T d \geq g^t d, \forall d$, which implies $\nabla f(x) = g$. ■

Example 2. Let $f(x) = |x|$, then $\partial f(0) = [-1, 1]$.
Proof: This is because $|x| \geq 0 + gx, \forall g \in [-1, 1]$. ■

Example 3. Let $f(x) = \|x\|_2$, then $\partial f(x) = \begin{cases} \frac{x}{\|x\|_2}, & x \neq 0 \\ \{g : \|g\|_2 \leq 1\}, & x = 0 \end{cases}$.
Proof: This is because $\|x\|_2 \geq 0 + g^T x, \forall \|g\|_2 \leq 1$. ■

Proposition 3.9 *If $\bar{x} \in int(dom(f))$, then $\partial f(\bar{x})$ is nonempty, closed, bounded, and convex.*

Proof:

- (Convexity and closedness): this is due to the fact that

  $$\partial f(\bar{x}) = \cap_x \{g : f(x) \geq f(\bar{x}) + g^T(x - \bar{x})\}$$

  is a infinite system of linear inequalities. The sub-differentiable set can be treated as the intersection of halfspaces, hence is closed and convex.

- (Non-emptyness): applying the separation theorem on $(\bar{x}, f(\bar{x}))$ and $epi(f) = \{(x, t) : f(x) \leq t\}$, we have

  $$\exists a, \beta, \text{s. t. } a^T \bar{x} + \beta f(\bar{x}) \leq a^T x + \beta t, \forall (x, t) \in epi(f).$$

  Claim: $\beta > 0$. We can first rule out $\beta \neq 0$ since $\bar{x} \in int(dom(f))$. We then rule out $\beta < 0$ by setting $x = \bar{x}$ and $t > f(\bar{x})$.
  Therefore, defining $g = \beta^{-1} a$, we have $f(x) \geq f(\bar{x}) + g^T(x - \bar{x})$, i.e. $g \in \partial(f)$.

- (Boundedness): if $\partial f(\bar{x})$ is unbounded, then there exist $s_k \in \partial f(\bar{x})$, such that $\|s_k\|_\infty \to \infty$ as $n \to \infty$. SInce $\bar{x} \in int(dom(f))$, there exits $\epsilon > 0$, such that $B(\bar{x}, \epsilon) = \{x : \|x - \bar{x}\| \leq \epsilon\} \subset dom(f)$. Hence, letting $y_k = \bar{x} + \epsilon \frac{s_k}{\|s_k\|}$, we have $y_k \in B(\bar{x}, \epsilon)$, and

  $$f(y_k) \geq f(\bar{x}) + s_k^T(y_k - \bar{x}) = f(\bar{x}) + \epsilon\|s_k\| \to \infty, \text{ as } k \to \infty.$$

  However, every convex function can be shown to be continuous on its interior; it is Lipschitz continuous on any convex compact subset on the domain. This implies that $f(x)$ is bounded on the compact ball $B(\bar{x}, \epsilon)$, which leads to a contradiction.

■

Remark. The reverse is also true. If $\forall x \in int(dom(f))$, $\partial f(x)$ is nonempty, then f is convex.

Proof: Let $x, y \in dom(f)$, $z = \lambda x + (1 - \lambda)y \in int(dom(f))$, we have

$$f(x) \geq f(z) + g^T(x - z)$$
$$f(y) \geq f(z) + g^T(y - z)$$

Hence, $\lambda f(x) + (1 - \lambda)f(y) \geq f(z) = f(\lambda x + (1 - \lambda)y)$. ■

3.6 Calculus of Sub-differential

Determining the subdifferentiable set of a convex function at a given point is in general very difficult. That's why calculus of subdifferentiable sets is particularly important in convex analyis.

1. **Taking conic combination**: If $h(x) = \lambda f(x) + \mu g(x)$, where $\lambda, \mu \geq 0$ and f, g are both convex, then

   $$\partial h(x) = \lambda \partial f(x) + \mu \partial g(x), \forall x \in int(dom(h)).$$

2. **Taking affine composition**: If $h(x) = f(Ax + b)$, where f is convex, then

   $$\partial h(x) = A^T \partial f(Ax + b).$$

3. **Taking supremum**: If $h(x) = \sup_{\alpha \in \mathcal{A}} f_\alpha(x)$ and each $f_\alpha(x)$ is convex, then

   $$\partial h(x) \supseteq conv\{\partial f_\alpha(x) | \alpha \in \alpha(x)\}$$

   where $\alpha(x) := \{\alpha : h(x) = f_\alpha(x)\}$.

4. **Taking superposition**: If $h(x) = F(f_1(x), \ldots, f_m(x))$, where $F(y_1, \ldots, y_m)$ is non-decreasing and convex, then

   $$\partial h(x) \supseteq \left\{ \sum_{i=1}^m d_i \partial f_i(x) : (d_1, \ldots, d_m) \in \partial F(y_1, \ldots, y_m) \right\}.$$

Example 1. Let $h(x) = \max_{1 \leq i \leq n}(a_i^T x + b_i)$, then $a_k \in \partial h(x)$ if k is some index such that $h(x) = a_k^T x + b_k$.

Example 2. Let $h(x) = \mathbb{E}[F(x, \xi)]$ be a convex function, then $g(x) = \int G(x, \xi) p(\xi) d\xi \in \partial h(x)$ if $G(x, \xi) \in \partial F(x, \xi)$ for each ξ.

Example 3. Let $h(x) = \max_{y \in C} f(x, y)$ where $f(x, y)$ is convex in x for any y and C is closed, then $\partial f(x, y_*(x)) \subset \partial h(x)$, where $y_*(x) = argmax_{y \in C} f(x, y)$.

This is because if $g \in \partial f(x, y_*(x))$, we have

$$h(z) \geq f(z, y_*(x)) \geq f(x, y_*(x)) + g^T(z - x) = h(z) + g^T(z - x).$$

3.7 Other Properties of Convex Functions

Jensen's inequality. Let f be a convex function, then

$$f(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i f(x_i)$$

as long as $\lambda_i \geq 0, \forall i$ and $\sum_i \lambda_i = 1$.

Moreover, let f be a convex function and X be a random variable, then

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

Example . The Kullback-Liebler distance between two distributions is nonnegative: i.e.

$$KL(p||q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right) \geq 0$$

where $p_i \geq 0, q_i \geq 0, \sum_i p_i = \sum_i q_i = 1$.

Proof: Let $f(x) = -\log(x)$, f is convex, so

$$-\log(\sum_i p_i x_i) \leq -\sum_i p_i \log(x_i).$$

Plugging $x_i = q_i/p_i$, this leads to

$$0 = -\log(\sum_i q_i) \leq \sum_i p_i \log(p_i/q_i).$$

■

References

[BV04] Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[BN01] Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, (Vol. 2). SIAM.

IE 598: Big Data Optimization **Fall 2016**

Lecture 5: Convex Optimization – September 6

Lecturer: Niao He *Scriber: Heran Zhang*

Overview In the introductory lecture, we discussed why convex optimization is a particularly interesting family of optimization problems to consider both from theoretical and practical viewpoints. In this lecture, we formally discuss what is a convex optimization problem and special properties that make it differ from general optimization problem. Moreover, we will study the optimality conditions for both unconstrained and constrained problems.

5.1 Basics of Convex Optimization

Recall that a canonical form of an optimization problem is

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \le 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, k \\ & x \in X \end{array} \tag{P}$$

Definition 5.1 (Convex Program) *A program (P) is convex if:*

(i) f *is convex;*

(ii) $g_i, i = 1, ..., m$ *are convex.*

(iii) there is either no equality constraints or only linear equality constraints

(iv) $X \subseteq dom(f) \bigcap \left(\cap_{i=1,\dots,m} dom(g_i)\right)$ *is a convex set.*

For the sake of simplicity, in the following we denote a convex program as

$$\boxed{\begin{array}{rl} \min & f(x), \\ \text{s.t.} & x \in S \\ S = \{x \in X : & g_i(x) \le 0, i = 1, \dots, m\} \end{array}}$$

Example 1 (LASSO) Recall the regularized least square regression model:

$$\min_w ||Xw - y||_2^2 + \lambda ||w||_1$$

Example 2 (Max-margin classification) Recall the classification model

$$\max_{w,b} \frac{2c}{||w||_2}, \ s.t. \ y_i(w^T x_i + b) \ge c, i = 1, \dots, n$$

This is equivalent as

$$\min_{w,b} ||w||_2, \ s.t. \ y_i(w^T x_i + b) \ge c, i = 1, \dots, n$$

which is a convex program.

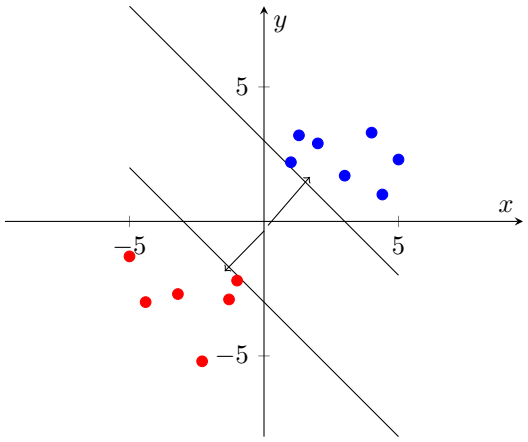


Figure 5.1: Max-margin classification

Definition 5.2 (Feasibility) *A solution x is feasible if x satisfies all constraints, i.e., $x \in S$. If $S = \emptyset$, we say program (P) is infeasible.*

Definition 5.3 (Optimality) *A solution x_* is optimal if x_* is feasible and $f(x_*) \leq f(x), \forall x \in S$. Usually, we denote as $x_* \in argmin_{x \in S} f(x)$.*

Definition 5.4 (Optimal value) *The optimal value is defined as $f_* = \inf_{x \in s} f(x)$*

- *Conventionally, if (p) is infeasible, we set $f_* = +\infty$.*
- *We say (P) is unbounded below if $f_* = -\infty$.*
- *We say (P) is solvable, if it has an optimal solution x_* and $f_* = f(x_*)$.*

Remark. The set of optimal solution, $\text{argmin}_{x \in S} f(x)$ is a convex set.

Remark. If f is strictly convex, then the optimal solution is unique.
Proof: Suppose there exist two optimal solutions x_1 and x_2, i.e. $f(x_1) = f(x_2) = f_*$. By strictly convexity, we have for $\lambda \in (0, 1)$,

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) f(x_2) = \lambda f_* + (1-\lambda) f_* = f_*$$

Since $\lambda x_1 + (1-\lambda)x_2 \in S$ is also feasible, this contradicts with the fact that f_* is the optimal value.

■

5.2 Local = Global

Definition 5.5 (Local minimum) *We say $x_* \in S$ is a local minimum if it has the smallest objective around its neighborhood, i.e. $\exists r > 0$, s.t. $f(x_*) \leq f(x)$, $\forall x \in B(x_*, r) \cap S$.*

Proposition 5.6 *A local minimum is a global minimum.*

Proof: Let x_* be a local minimum. For any $x \in S$, when ϵ is small enough, we have $y = x_* + \epsilon(x - x_*) \in B(x_*, r) \cap S$ and $0 \leq \frac{f(y)-f(x_*)}{\|y-x_*\|} \leq \frac{f(x)-f(x_*)}{||x-x_*||} \Rightarrow f(x) \geq f(x_*), \forall x \in S$. ■

Note that in general, for any optimization (possibly non-convex) problems, this is not true. A local minimum is not necessarily global optimum. This is a key property that makes convex optimization problem different from general non-convex problem.

5.3 Optimality Conditions for Simple Constrained Problems

We consider the simple constrained optimization problem (no inequality constraints)

$$\begin{array}{|ll|}\hline \min & f(x) \\ \text{s.t.} & x \in X \\ \hline \end{array}$$

5.3.1 Differentiable Case

Theorem 5.7 (Sufficient and Necessary Condition) *Assume f is convex and differentiable on $x_* \in X$,*

$$x_* \text{ is an optimal solution } \iff (x - x_*)^T \nabla f(x_*) \geq 0, \forall x \in X.$$

Proof:

- ($\Leftarrow$) Since f is convex, we have $\forall x \in X$, $f(x) \geq f(x_*) + \nabla f(x_*)^T(x - x_*) \geq f(x_*)$.
- ($\Rightarrow$) We have $(x - x_*)^T \nabla f(x_*) = \lim_{\epsilon \to 0} \frac{f(x_* + \epsilon(x - x_*)) - f(x_*)}{\epsilon} \geq 0$.

■

Corollary 5.8 (Unconstrained Case) *If $x_* \in int(X)$, then x_* is optimal if and only if $\nabla f(x_*) = 0$. In particular, if $X = \mathbb{R}^n$, x_* is optimal if and only if $\nabla f(x_*) = 0$.*

Remark. Note that for unconstrained convex problems, $\nabla f(x_*) = 0$ is both sufficient and necessary for x_* to be optimal. However, for general (non-convex) optimization problems, $\nabla f(x_*)$ is only a necessary condition and is not sufficient to guarantee the global optimality. In fact, if $\nabla f(x_*) = 0$, we call x_* a stationary point of f, which could be a local minimum, a local maximum, or a saddle point.

Example 3 (Quadratic Problem) $f(x) = \frac{1}{2} x^T Q x + b^T x + c,\ Q \succeq 0$.
The optimal solution satisfies $\nabla f(x_*) = Q x_* + b = 0$.

$$\underset{x \in \mathbb{R}^n}{\text{argmin}} f(x) = \begin{cases} -Q^{-1}b & \text{Q is nonsingular} \\ \emptyset & \text{Q is singular, } b \notin col(Q) \\ x_* + null(Q) & \text{Q is singular, } b \in col(Q) \end{cases}$$

where x_* is such that $Q x_* + b = 0$ and $null(Q) = \{d : Qd = 0\}$.
Note that the set of optimal solution might be an empty set, or a unique solution, or a convex set.

5.3.2 Non-differentiable Case

Let us consider the situation when f is not necessarily differentiable.

Proposition 5.9 (Sufficient and Necessary Condition) *Assume f is convex on X, then*

(a) $x_* \in X$ is optimal if and only if $\exists g \in \partial f(x_*)$, s.t. $g^T(x - x_*) \geq 0, \forall x \in X$.

(b) Suppose $X = \mathbb{R}^n$, x_* is optimal if and only if $0 \in \partial f(x_*)$.

Proof:

(a) "If" part : by definition of subgradient, we have $\forall x \in X, f(x) \geq f(x_*) + g^T(x - x_*) \geq f(x_*)$.
"Only if" part: this uses the fact that

$$\lim_{\epsilon \to 0} \frac{f(x + \epsilon d) - f(x)}{\epsilon} = \sup_{g \in \partial f(x)} g^T d.$$

We therefore have $\sup_{g \in \partial f(x_*)} g^T(x - x_*) \geq 0$. Since $\partial f(x_*)$ is closed, this implies that there exists $g \in \partial f(x_*)$ such that $g^T(x - x_*) \geq 0$.

(b) This is because

$$f(x) \geq f(x_*) \Longleftrightarrow f(x) \geq f(x_*) + 0^T(x - x_*) \Leftrightarrow 0 \in \partial f(x_*)$$

■

5.4 Optimality Condition for Constrained Problem

We now consider the constrained optimization problem

$$\begin{array}{|ll|} \hline \min & f(x) \\ \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & x \in X \\ \hline \end{array}$$

5.4.0 Motivation

A solution x_* is optimal if and if the following holds true:

1. The system $\{x \in X : f(x) \leq f(x_*), g_i(x) \leq 0, i = 1, ..., m\}$ has a solution
2. The system $\{x \in X : f(x) < f(x_*), g_i(x) \leq 0, i = 1, ..., m\}$ has no solution

One verifiable condition to show that the second system has no solution is when there exists nonnegative coefficients $\lambda_i \geq 0, i = 1, \ldots, m$, such that the system $\{x \in X : f(x) + \sum \lambda_i g_i(x) < f(x_*)\}$ has no solution. This is equivalent as $\inf_{x \in X} f(x) + \sum \lambda_i g_i(x) \leq f(x_*)$. In fact, under some condition, the opposite direction is also true. This is due to the useful convex theorem of alternative, which we present below without detailing the proof. The proof follows from the Separation theorem.

Definition 5.10 (Slater Condition) *Problem (P) is said to satisfy the Slater condition if there exists* $x \in X$ *such that* $g_i(x) < 0,\ \forall i = 1, ..., m$.

Theorem 5.11 (Convex Theorem of Alternatives) *Consider the two systems*

$$(I) : \{x \in X : f(x) < c, g_i(x) \leq 0, i = 1, ..., m\}$$

$$(II) : \{\lambda \geq 0 : \inf_{x \in X} f(x) + \sum \lambda_i g_i(x) \geq c\}$$

Then (I) *is insolvable if and only if* (II) *is solvable.*

5.4.1 Lagrangian Duality

We define

- $L(x, \lambda) = f(x) + \sum \lambda_i g_i(x)$ (Lagrangian function)
- $\underline{L}(\lambda) = \inf_{x \in X} L(x, \lambda)$ (Lagragian dual function)

Primal:

$$\begin{aligned} \text{Opt}(P) = \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \ \ i = 1, \ldots, m \\ & x \in X \end{aligned}$$

Lagrangian dual:

$$\text{Opt}(D) = \sup_{\lambda \geq 0} \underline{L}(\lambda)$$

$$\text{where } \underline{L}(\lambda) = \inf_{x \in X} L(x, \lambda)$$

Theorem 5.12 (Duality)

1. **(Weak duality)** $\text{Opt}(D) \leq \text{Opt}(P)$
2. **(Strong duality)** *If* (P) *is solvable and satisfies slater condition, then* (D) *is solvable and* $\text{Opt}(D) = \text{Opt}(P)$.

Proof:

1. $\forall \lambda \geq 0$, we have

   $$\underline{L}(\lambda) = \inf_{x \in X} L(x, \lambda) \leq \inf_{x \text{ feasible}} L(x, \lambda) \leq \inf_{x \text{ feasible}} f(x) = \text{Opt}(P)$$

   Hence, $\text{Opt}(D) = \sup_{\lambda \geq 0} \underline{L}(\lambda) \leq \text{Opt}(P)$.

2. If x_* is optimal for (P), then the system $\{x \in X : f(x) < f(x_*), g_i(x) \leq 0, i = 1, ..., m\}$ has no solution. From the convex theorem of alternative, this implies that there exists $\lambda \geq$, such that $\underline{L}(\lambda) \geq \text{Opt}(P)$. Hence, $\text{Opt}(D) \geq \text{Opt}(P)$.

■

IE 598: Big Data Optimization **Fall 2016**

Lecture 6: Conic Optimization – September 8

Lecturer: Niao He *Scriber: Juan Xu*

Overview In this lecture, we finish up our previous discussion on optimality conditions for constrained convex programs. We will then introduce an important class of well-structured convex optimization problems – conic programming.

6.1 Recap

In the previous lecture, we have discussed the optimality condition for simple constrained or unconstrained convex problems

- If $f(x)$ is a convex and differentiable function defined on a convex set X, then

$$x_* \in \arg\min_{x\in X} f(x) \quad \Leftrightarrow \quad \nabla f(x_*)^\top (x - x_*) \geq 0 \quad \forall x \in X,$$

- If $f(x)$ is a convex and differentiable function defined on $\mathbb{R}^n$, then

$$x_* \in \arg\min_{x\in \mathbb{R}^n} f(x) \quad \Leftrightarrow \quad \nabla f(x_*) = 0.$$

What about constrained convex problems?

$$\begin{aligned} \textbf{Primal:} \quad \text{Opt(P)} = \quad & \min f(x) \\ & \text{s.t. } g_i(x) \leq 0 \quad \forall i = 1, \ldots, m, \\ & x \in X. \end{aligned}$$

$$\begin{aligned} \textbf{Lagrangian dual:} \quad \text{Opt(D)} = \quad & \sup_{\lambda \geq 0} \underline{\mathrm{L}}(\lambda) \\ & \text{where } \underline{\mathrm{L}}(\lambda) = \inf_{x\in X} \{f(x) + \sum_{i=1}^m \lambda_i g_i(x)\}. \end{aligned}$$

Recall the Lagrangian duality theorem,

- Weak duality: Opt(D)$\leq$Opt(P);
- Strong duality: Under slater condition, Opt(P)=Opt(D).

Solving the primal problem is essentially the same as solving its Lagrangian dual problem.

6.2 Constrained Convex Optimization (cont'd)

6.2.1 Saddle Point Formulation

Recall the Lagrangian function:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x).$$

We can rewrite the primal and its Lagrangian dual problems in the following form.

$$\text{Opt(D)} = \sup_{\lambda \geq 0} \underbrace{\inf_{x \in X} L(x, \lambda)}_{:= \underline{L}(\lambda)}$$

$$\text{Opt(P)} = \inf_{x \in X} \underbrace{\sup_{\lambda \geq 0} L(x, \lambda)}_{:= \overline{L}(x)}$$

Note that $\overline{L}(x) = \begin{cases} f(x) & g_i(x) \leq 0, \forall i \\ +\infty & \text{otherwise} \end{cases}$. Hence, $\inf_{x \in X} \overline{L}(x) = \text{Opt(P)}$.

Definition 6.1 ***(Saddle Point)***
We call (x_*, λ^*), where $x_* \in X$ and $\lambda^* \geq 0$ a saddle point of $L(x, \lambda)$ if and only if $L(x, \lambda^*) \geq L(x_*, \lambda^*) \geq L(x_*, \lambda), \forall x \in X, \forall \lambda \geq 0$.

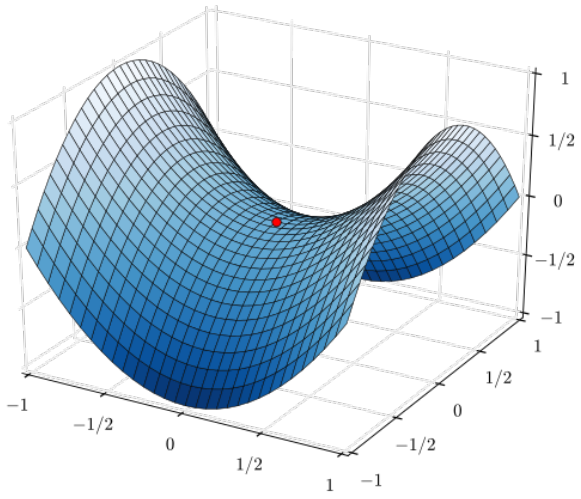


Figure 6.1: Saddle point (Source: https://en.wikipedia.org/wiki/Saddle_point)

Remark. From Figure (6.1), we can see that the saddle point (the red point) is the local minimum point in the x-axis direction, and the local maximum point in the y-axis direction.

Theorem 6.2 *(x_*, λ^*) is a saddle point of $L(x, \lambda)$ if and only if x_* is optimal for (P), λ^* is optimal for (D), and Opt(P)=Opt(D).*

Proof:

- ($\Rightarrow$): On the one hand, we have

$$\text{Opt}(D) = \sup_{\lambda \geq 0} \inf_{x \in X} L(x, \lambda) \geq \inf_{x \in X} L(x, \lambda^*) = L(x_*, \lambda^*).$$

  The last equality is due to the fact that $L(x, \lambda^*) \geq L(x_*, \lambda^*), \forall x \in X$. On the other hand, we have

$$\text{Opt}(D) = \inf_{x \in X} \sup_{\lambda \geq 0} L(x, \lambda) \leq \sup_{\lambda \geq 0} L(x_*, \lambda) = L(x_*, \lambda^*).$$

  The last equality is due to the fact that $L(x_*, \lambda^*) \geq L(x_*, \lambda), \forall \lambda \geq 0$. We arrive at Opt(D)$\geq L(x_*, \lambda^*) \geq$ Opt(P), and by weak duality, Opt(D)$\leq$Opt(P)); it can then be shown that Opt(P)=Opt(D), x_* is optimal for (P), and λ^* is optimal for (D).

- ($\Leftarrow$): Since x_* is optimal for (P), λ^* is optimal for (D), we have

$$\text{Opt}(D) = \underline{L}(\lambda^*) = \inf_{x \in X} L(x, \lambda^*) \leq L(x_*, \lambda^*)$$

$$\text{Opt}(P) = \overline{L}(x_*) = \sup_{\lambda \geq 0} L(x_*, \lambda) \geq L(x_*, \lambda^*)$$

  Now that $\text{Opt}(P) = \text{Opt}(D)$, we must have $\text{Opt}(P) = L(x_*, \lambda^*) = \text{Opt}(D)$. That is,

$$\inf_{x \in X} L(x, \lambda^*) = L(x_*, \lambda^*) = \sup_{\lambda \geq 0} L(x_*, \lambda)$$

  which implies that (x_*, λ^*) is a saddle point.

■

6.2.2 Optimality conditions for constrained convex problem

Theorem 6.3 ***(Saddle point condition)***
If (P) is a convex optimization problem, and x_* is a feasible solution to (P), then

$$x_* \text{ is optimal for (P)} \overset{\text{under slater condition}}{\Longleftrightarrow} \exists (x_*, \lambda^*) \text{ is a saddle point of } L(x, \lambda) \tag{6.1}$$

Proof: ($\Rightarrow$): Under the slater condition, if x_* is the optimal solution of (P), we can find an optimal solution λ^* for the Lagrangian dual problem such that Opt(P)=Opt(D). By Theorem (6.2), (x_*, λ^*) is a saddle point of $L(x, \lambda)$. ($\Leftarrow$): By Theorem (6.2), if (x_*, λ^*) is a saddle point of $L(x, \lambda)$, then x_* is optimal for (P). ■

Theorem 6.4 ***(Karush-Kuhn-Tucker (KKT) condition)***
If (P) is convex, x_* is a feasible solution to (P), and $f(x)$, $g_i(x)$'s are all differentiable at x_*, then

$$x_* \text{ is optimal for (P)} \overset{\text{under slater condition}}{\Longleftrightarrow} \exists \lambda^* \geq 0 \text{ s.t. } \begin{array}{ll} (a) & \nabla f(x_*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x_*) \in N_X(x_*); \\ (b) & \lambda_i^* g_i(x_*) = 0, \forall i = 1, \ldots, m \end{array} \tag{6.2}$$

where $N_X(x_*)$ represents the normal cone of x_* ($N_X(x_*) = \{g : g^\top (x - x_*) \geq 0, \forall x \in X\}$).

We refer to (a) as the Lagrangian stationarity condition, (b) the complementary slackness condition. Both (a) and (b) consist of the KKT condition.

Proof: The proof follows directly from the previous theorem. First,

$$L(x,\lambda^*) \geq L(x_*,\lambda^*), \quad \forall x \in X \quad \Leftrightarrow \quad \frac{\partial L}{\partial x}(x_*,\lambda^*)^\top (x - x_*) \geq 0, \quad \forall x \in X \Leftrightarrow \text{ condition } (a)$$

This is because $\frac{\partial L}{\partial x}(x_*,\lambda^*) = \nabla f(x_*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x_*))$. Second,

$$\begin{aligned} L(x_*,\lambda^*) \geq L(x_*,\lambda), \quad \forall \lambda \geq 0 \quad &\Leftrightarrow \quad f(x_*) + \sum_{i=1}^m \lambda_i^* g_i(x_*) \geq f(x_*) + \sum_{i=1}^m \lambda_i g_i(x_*) \quad \forall \lambda \geq 0 \\ &\Leftrightarrow \quad \sum_{i=1}^m \lambda_i^* g_i(x_*) \geq \sum_{i=1}^m \lambda_i g_i(x_*) \quad \forall \lambda \geq 0 \\ &\Leftrightarrow \quad \text{condition } (b) \quad (\text{because } g_i(x_*) \leq 0, \ \forall i = 1,\ldots,m, \text{ and } \lambda \geq 0). \end{aligned}$$

Therefore, (x_*,λ^*) is a saddle point if and only if (a) and (b) hold true. ■

6.3 Conic Optimization

6.3.1 A bit of history

Mid-1940s	Dantzig
	Simplex method for solving linear programming
	Worst cast exponential time (however, fast when implemented)
Mid-1970s	Shor, Khachiyan, Nemirovski, Yudin
	Ellipsoid method for linear programming and convex optimization
	Polynomial time (however, very slow for large problems)
	Complexity: $O(n^2 \log(\frac{1}{\epsilon})) \cdot O(n^2)$
Mid-1980s	Karmarkar, Nemirovski, Nesterov
	Interior point method for linear programming and well-structured convex problem
	Polynomial time
	Complexity: $O(\nu \cdot \log(\frac{\nu}{\epsilon})) \cdot O(n^3)$

The poor performance of some algorithms (e.g. Ellipsoid method) despite of polynomial time stems from their black box oriented nature - these algorithms do not adjust themselves to the structure of the problem and only utilize local information, e.g. the values and (sub)gradients of the objective and the constraints at query points. While in contrast, interior point method is designed to take advantage of the problem's structure and is much more efficient.

6.3.2 From linear programming to conic programming

When passing from a linear program to a nonlinear convex program, we can act as follows

$$\begin{aligned} \text{(Linear Programming)} \qquad & \min\{c^\top x : Ax - b \geq 0\} \quad (\text{“}\geq\text{” is component-wise}) \\ = \quad & \min\{c^\top x : Ax - b \in \mathbb{R}^n_+\}. \end{aligned}$$

$$\begin{aligned} \text{(Conic Programming)} \qquad & \min\{c^\top x : Ax - b \geq_{\mathbf{K}} 0\} \quad (\text{“}\geq_{\mathbf{K}}\text{” is some order}) \\ = \quad & \min\{c^\top x : Ax - b \in \mathbf{K}\}, \end{aligned}$$

where $\mathbf{K}$ is a regular (closed, pointed, nonempty interior) cone. The order “$\geq_{\mathbf{K}}$” is defined as $x \geq_{\mathbf{K}} y$ if and only if $x - y \in \mathbf{K}$. It is easy to see that linear programming is a special case of conic programming which will be shown below.

6.3.3 Three typical conic problems

1. **Linear programming (LP)**
   $\mathbf{K} = \mathbb{R}^{\mathbf{n}}_+ = \mathbb{R}_+ \times \cdots \times \mathbb{R}_+$.

   $$\textbf{(LP):} \quad \min\{c^\top x : a_i^\top x - b_i \geq 0, i = 1, \ldots, k\}$$

2. **Second-order cone programming (SOCP, also called conic quadratic programming)**
   $\mathbf{K} = \mathbf{L^{n_1}} \times \cdots \times \mathbf{L^{n_k}}$, where $\mathbf{L^n} = \{(\nu_{\mathbf{1}}, \ldots, \nu_{\mathbf{n}}),\ \nu_{\mathbf{n}} \geq \sqrt{\nu_{\mathbf{1}}^{\mathbf{2}} + \ldots \nu_{\mathbf{n-1}}^{\mathbf{2}}}\}$.

   $$\textbf{(SOCP):} \quad \min\{c^\top x : ||A_i x - b_i||_2 \leq c_i^\top x - d_i, i = 1, \ldots, k\}$$

   One can treat $Ax - b = \left[A_1 x - b_1, c_1^\top x - d_1; \cdots; A_k x - b_k, c_k^\top x - d_k\right]$.

3. **Semi-definite programming (SDP)**
   $\mathbf{k} = \mathbf{S^{n_1}_+} \times \cdots \times \mathbf{S^{n_k}_+}$, where $\mathbf{S^n_+} = \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{X} = \mathbf{X}^\top\}$.

   $$\textbf{(SDP):} \quad \min\{c^\top x : A_i x - b_i \succeq 0, i = 1, \ldots, k\};$$

   This can be simplified as

   $$\textbf{(SDP):} \quad \min\{c^\top x : \mathcal{A}x - b \succeq 0\},$$

   because

   $$Ax - b := \begin{bmatrix} A_1 x - b_1 & & \\ & \ddots & \\ & & A_k x - b_k \end{bmatrix} \succeq 0 \Leftrightarrow A_i x - b_i \succeq 0,\ \forall i = 1, \ldots, k.$$

Lemma. (Schur complement for symmetric positive semidefinite matrices)
If $S = \begin{bmatrix} P & R^\top \\ R & Q \end{bmatrix}$ is symmetric with $Q \succ 0$, then $S \succeq 0 \Leftrightarrow P - R^\top Q^{-1} R \succeq 0$.

Remark. We show that **(LP)**$\subseteq$**(SOCP)**$\subseteq$**(SDP)**.

- **(LP)⊆(SOCP)**

$$a_i^\top x - b_i \geq 0 \Leftrightarrow \begin{bmatrix} 0 \\ a_i^\top x - b_i \end{bmatrix} \geq_{\mathbf{L^2}} 0.$$

- **(SOCP)⊆(SDP)**

$$\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \in \mathbf{L^m} \Leftrightarrow \begin{bmatrix} x_m & x_1 & \cdots & x_{m-1} \\ x_1 & x_m & & \\ \vdots & & \ddots & \\ x_{m-1} & & & x_m \end{bmatrix} \succeq 0 \xLeftrightarrow{\textbf{by Schur lemma}} x_m \geq \sqrt{x_1^2 + \cdots + x_{m-1}^2}.$$

Note that the above matrix is linear in x and can be written as $\sum_i x_i A_i$ with properly defined A_i.

6.3.4 Calculus of Conic Programming

Let $\mathcal{K}$ be a family of regular cones, e.g., $\mathbb{R}^n_+, \mathbf{L^n}, \mathbf{S^n_+}$, or their inner points.

Definition 6.5 *X is $\mathcal{K}$-representable if $\exists \mathbf{K} \in \mathcal{K}$, s.t. $X = \{x : \exists u, Ax + Bu - b \in \mathbf{K}\}$.*

Definition 6.6 *$f(x)$ is $\mathcal{K}$-representable if $epi(f)$ is $\mathcal{K}$-representable, i.e., $\exists \mathbf{K} \in \mathcal{K}$, s.t.,*

$$epi(f) = \{(x,t) : \exists v, Cx + dt + Dv - e \in \mathbf{K}\}.$$

Remark It can be shown that many convexity-preserving operations preserve $\mathcal{K}$-representability, e. g

- taking intersections, direct product, affine image of $\mathcal{K}$-representable sets;
- taking conic combination, affine composition, partial minimization of $\mathcal{K}$-representable functions.

Here we provide some examples:

Examples (SOCP-r functions)

- $f(x) = ||x||_2$
  $epi(f) = \{(x,t) : t \geq ||x||_2\} \Leftrightarrow \{(x,t) : \begin{bmatrix} x \\ t \end{bmatrix} \in \mathbf{L^{n+1}}\}.$
- $f(x) = x^\top x$
  $epi(f) = \{(x,t) : t \geq x^\top x\} \Leftrightarrow \{(x,t) : (t+1)^2 \geq (t-1)^2 + 4x^\top x\} \Leftrightarrow \{(x,t) : \begin{bmatrix} 2x \\ t-1 \\ t+1 \end{bmatrix} \in \mathbf{L^{n+2}}\}.$
- $f(x) = x^\top Q x + q^\top x + t$, where $Q = L \cdot L^\top$
  $epi(f) = \{(x,t) : t \geq x^\top Q x + q^\top x + t\} \Leftrightarrow \{(x,t) : \begin{bmatrix} 2L^\top x \\ t - q^\top x - r - 1 \\ t - q^\top x - r + 1 \end{bmatrix} \in \mathbf{L^{n+2}}\}.$

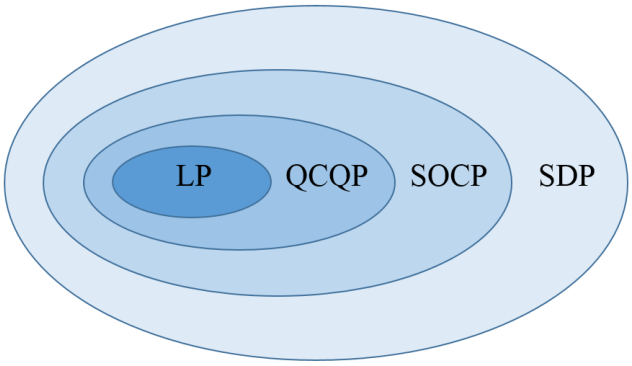


Figure 6.2: Relationships among LP, QCQP, SOCP, SDP

(QCQP refers to quadratically constrained quadratic programming.)

References

[BV04] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization.* Cambridge university press.

[DN01] Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, (Vol. 2). SIAM.

IE 598: Big Data Optimization **Fall 2016**

Lecture 7: Introduction to Optimization Algorithms – September 13

Lecturer: Niao He *Scriber: Shripad Gade*

Overview: In this lecture we conclude the discussion on Conic Programming. We revisited the definitions of $\mathcal{K}$ representable functions (and sets), and discussed its applications in rewriting convex optimization problems into conic programs. We also discussed duality in conic programs followed by a brief introduction to Interior Point Methods. We started a discussion on taxonomy for optimization algorithms along with the notion of error in optimization.

7.1 Conic Programming

7.1.1 Recap

Given a cone $\mathbf{K}$ in $\mathbb{R}^m$ (convex, pointed, closed with nonempty interior), an objective $c \in \mathbb{R}^n$, a $m \times n$ constraint matrix A with right hand side $b \in \mathbb{R}^m$, a conic program is defined as -

$$\min_x \{c^T x \mid Ax - b \geq_{\mathbf{K}} 0\}. \tag{7.1}$$

A few special types of conic programs -

- (Non-negative Orthant) $\mathbf{K} = \mathbb{R}^m_+$, the constraint will become $Ax - b \geq 0$. The resulting conic program is a Linear Program (LP).
- (Lorentz Cone) $\mathbf{K} = \mathcal{L}^m$, the constraint can be written in the form of $\|Ax - b\|_2 \leq c^T x - d$. This gives a Second Order Cone Program (SOCP), also called Conic Quadratic Program (CQP).
- (Semi-definite Cone) $\mathbf{K} = S^m_+$, the constraint is $\sum x_i A_i - b \succcurlyeq 0$. The resulting program is a Semi-definite Program (SDP).

As proved in Lecture 6, LP $\subseteq$ SOCP $\subseteq$ SDP.

Let $\mathcal{K}$ be a family of regular cones closed with taking direct product.

Definition 7.1 ($\mathcal{K}$ Representable Sets) *A set X is called conic representable or $\mathcal{K} - r$ if $\exists\ \boldsymbol{K} \in \mathcal{K}$ such that, $X = \{x : \exists u, Ax + Bu - b \in \boldsymbol{K}\}$.*

Definition 7.2 ($\mathcal{K}$ Representable Functions) *A function $f(x)$ is $\mathcal{K} - r$ if $epi(f) = \{(x,t) : t \geq f(x)\}$ is a $\mathcal{K} - r$ set, i.e. $\exists\ \boldsymbol{K} \in \mathcal{K}$ such that, $epi(f) = \{(x,t) : \exists v, Cx + dt + Dv - e \in \boldsymbol{K}\}$*

Using the above definitions we can claim that if $f(x)$ is $\mathcal{K} - r$ function and X is a $\mathcal{K} - r$ set, then any optimization programs can be equivalently written as,

$$\begin{array}{lll} \min\limits_{x} & f(x) & \\ s.t. & x \in X & \end{array} \iff \begin{array}{lll} \min\limits_{x,t} & t & \\ s.t. & t \geq f(x) & \\ & x \in X & \end{array} \iff \begin{array}{lll} \min\limits_{x,t,u,v} & t & \\ s.t. & Cx + dt + D\nu - e \in \mathbf{K}_f & \\ & Ax + Bu - b \in \mathbf{K}_X & \end{array}$$

Examples of SOCP-r functions

- $f(x) = \|x\|_2$, we have $epi(f) = \{(x,t) : t \geq \|x\|_2\} \iff \{(x,t) : \begin{bmatrix} x \\ t \end{bmatrix} \in \mathcal{L}^{n+1}\}$
- $f(x) = x^T x$, we have $epi(f) = \{(x,t) : t \geq x^T x\} \iff \begin{bmatrix} 2x \\ t-1 \\ t+1 \end{bmatrix} \in \mathcal{L}^{n+2}$
- $f(x) = x^T Q x + q^T x + r$ where $Q = LL^T \succcurlyeq 0$

  $epi(f) = \{(x,t) : t \geq x^T Q x + q^T x + r\} \iff \left\{ (x,t) : \begin{bmatrix} 2L^T x \\ t - q^T x - r - 1 \\ t - q^T x - r + 1 \end{bmatrix} \in \mathcal{L}^{n+2} \right\}$
- $f(x) = x^2 + 2x^4$, we have

$$epi(f) = \left\{ (x,t) : \exists\, t_1, t_2 \text{ s.t. } \begin{bmatrix} 2x \\ t_1 - 1 \\ t_1 + 1 \end{bmatrix} \in \mathcal{L}^3, \begin{bmatrix} 2t_1 \\ t_2 - 1 \\ t_2 + 1 \end{bmatrix} \in \mathcal{L}^3, \begin{bmatrix} 0 \\ t - t_1 - 2t_2 \end{bmatrix} \in \mathcal{L}^2 \right\}$$

Examples of SDP-r functions

- $f(X) = \lambda_{\max}(X)$ on S_+^m.
  Note, $t \geq \lambda_{\max}(X) \iff tI_m - X \succcurlyeq 0$.
- $f(X) = \sigma_{\max}(X) = \max_{\|u\|_2 \leq 1} \|Xu\|_2$ on $\mathbb{R}^{m \times n}$.
  Note, $t \geq \sigma_{\max}(X) \iff t \geq \lambda_{\max}(X^T X) \iff t^2 I - X^T X \succcurlyeq 0 \underset{Schur}{\iff} \begin{bmatrix} tI_n & X^T \\ X & tI_m \end{bmatrix} \succcurlyeq 0.$

Example - [Group Lasso] We can show that sparse group lasso

$$\min_w \left\{ \|Xw - y\|_2^2 + \lambda \sum_{i=1}^{p} \|w_i\|_2 \right\}$$

is a SOCP problem. We can reformulate this problem into a SOCP as follows -

$$\min \quad t_0 + \lambda \sum_{i=1}^{k} t_i \tag{7.2}$$

$$s.t. \quad \begin{bmatrix} 2(Xw - y) \\ t_0 - 1 \\ t_0 + 1 \end{bmatrix} \in \mathcal{L}^{m+2}, \ \begin{bmatrix} w_i \\ t_i \end{bmatrix} \in \mathcal{L}^{n_i+1} \ \forall i \tag{7.3}$$

7.1.2 Conic Problem - Duality

Duality in conic programs is simple. We can write a conic program and its dual as follows:

$$\text{CP (P)}: \quad \min_x \{c^T x | Ax - b \succcurlyeq_{\mathbf{K}} 0\}$$

$$\text{CP Dual (D)}: \quad \max_y \{b^T y | A^T y = c, y \succcurlyeq_{\mathbf{K}_*} 0\}$$

The dual program is defined using partial ordering on the dual cone $\mathbf{K}_*$. The dual cone comprises of vectors whose inner product with all vectors of the cone $\mathbf{K}$ are non-negative. The definition can be formally written as:

Definition 7.3 *Let $\boldsymbol{K} \in \mathcal{K}$ be a nonempty set. Then the dual cone $\boldsymbol{K}_*$ is given as,*

$$\boldsymbol{K}_* = \{y : y^T z \geq 0,\ \forall z \in \boldsymbol{K}\}.$$

The dual cone has some interesting properties -

Remark 7.4 *If $\boldsymbol{K}$ is a closed convex cone, then so is $\boldsymbol{K}_*$.*

Remark 7.5 *The dual cone to $\boldsymbol{K}_*$ is $\boldsymbol{K}$ itself, i.e. $(\boldsymbol{K}_*)_* = \boldsymbol{K}$.*

Examples of dual cones for some family of cones -

- $(\mathbb{R}^m_+)_* = \mathbb{R}^m_+$
- $(S^m_+)_* = S^m_+$
- $(\mathcal{L}^m)_* = \mathcal{L}^m$

Recall the strong and weak duality theorems statements discussed in previous lectures.

- Weak Duality - $Opt(P) \geq Opt(D)$
- Strong Duality - If P is strictly feasible and bounded then (D) is solvable and $Opt(D) = Opt(P)$.

We are interested in obtaining similar results for CP and its dual problem. A weak duality statement can be obtained easily.

Theorem 7.6 (Weak Duality) *The optimal value of (D) is a lower bound on the optimal value of (P).*

Proof: If x is a feasible solution to CP then we have $A^T x \succcurlyeq_{\mathbf{K}} b$ (or $Ax - b \in \mathbf{K}$). And for an admissible vector y belonging to the dual cone ($y \in \mathbf{K}_*$) we get the scalar equation, $y^T(Ax - b) \geq 0 \implies y^T Ax \geq y^T b$.

If y is a feasible solution to the dual problem (D), then we have $A^T y = c$.

Clearly, using the conditions obtained above,

$$c^T x = (A^T y)^T x = y^T Ax \geq y^T b = b^T y$$

$$c^T x \geq b^T y$$

The condition derived above is valid for all feasible solutions of (P) and (D). Hence, the minimum value of $c^T x$ or the solution to the primal CP or (P) is lower bounded by the maximum value of $b^T y$ or the solution to the dual problem (D). [1] ■

A stronger argument can indeed be made for a conic program (P) and its dual (D) (see Conic Duality Theorem, (Theorem 2.4.1) in [BTN01]). The optimality conditions for the primal-dual solution are characterized by the following statement.

[1] A more inquisitive reader is directed to Section 2.3-2.4 of [BTN01] for a more thorough treatment of duality.

Theorem 7.7 (Optimality Conditions) *If one of (P) or (D) is strictly feasible then (x_*, y_*) is optimal solution to the Primal-Dual problem,*

- *if and only if $c^T x_* - b^T y_* = 0$ [Zero Duality Gap]*
- *if and only if $y_*^T (Ax_* - b) = 0$ [Complimentary Slackness]*

Proof: We begin by first proving that (x_*, y_*) is optimal solution is equivalent to the zero duality gap.

$$\begin{aligned} c^T x_* - b^T y_* &= c^T x_* - b^T y_* - Opt(P) + Opt(P) + Opt(D) - Opt(D) \\ &= \underbrace{\left[c^T x_* - Opt(P)\right]}_{\geq 0} + \underbrace{\left[Opt(D) - b^T y_*\right]}_{\geq 0} + \underbrace{Opt(P) - Opt(D)}_{\geq 0} \quad \ldots \text{Weak Duality, Theorem 7.6} \end{aligned}$$

The duality gap is always greater than or equal to zero, and it is exactly zero if and only if $Opt(P) = Opt(D)$. Clearly the duality gap being zero is necessary and sufficient for the primal-dual variables to be optimum.

We can obtain complimentary slackness condition directly from duality gap condition. We assume that the solution (x_*, y_*) is feasible and we get,

$$\begin{aligned} c^T x_* - b^T y_* &= (A^T y)^T x_* - b^T y_* \\ &= y^T \left[Ax_* - b\right] = 0 \qquad \ldots \text{if and only if } c^T x_* - b^T y_* = 0 \end{aligned}$$

Hence the complimentary slackness being zero is also necessary and sufficient for the primal-dual variable to be the optimum. ■

7.2 Interior Point Methods

Interior Point Methods are a class of algorithms that solve linear and nonlinear convex optimization problems [BV04,DT06,NT08]. Interior point method was first introduced by John von Neumann (in discussion with George Dantzig) in 1948 [DT06], however the method was inefficient and slower in practice as compared to the Simplex method. Karmarkar proposed a provable polynomial-time algorithm (called Karmarkar's algorithm) for linear programming problems in 1984 [K84]. It was much more efficient than Dantzig's simplex method and allowed solving a larger class of optimization problems. Karmarkar's improvements revitalized the study of interior point algorithms. And with further refinements, the Interior Point Methods have become extremely efficient and have found applications in virtually every field of continuous optimization.

7.2.1 Barrier Method

Interior point methods typically solve the constrained convex optimization problem by applying Newton's method to a sequence of equality constrained problems. We will study a particular interior point algorithm, the barrier method, in this lecture. Consider an optimization problem as shown below:

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & x \in X. \end{aligned} \tag{7.4}$$

Barrier methods as the name suggest employ barrier functions to integrate inequality constraints into the objective function. First, we motivate the definition and development of barrier functions.

Barrier function. Since we want to merge inequality constraints to the objective, the following conditions on barrier functions seem natural

- Barrier function should map the feasible set $\text{int}(X)$ in (7.4) to real space, i.e. $F : \text{int}(X) \to \mathbb{R}$.
- $F(x)$ is integrated into the objective function and we intend to use gradient based method, so the barrier function should be reasonably "smooth" (continuously differentiable).
- $F(x)$ should behave like a barrier and grow unbounded as the state (x) enters the boundary of the feasible set, i.e. $F(x) \to +\infty$ if $x \in \partial X$.
- The Hessian (or second derivative for functions of scalars) needs to be positive definite, $\nabla^2 F(x) \succ 0$, $\forall x \in X$.

One would, almost unconsciously, gravitate towards indicator functions as potential barrier functions to integrate the constraint in Problem (7.4) with the objective function (see Section 11.2 in [BV04]). However, the non-smooth and non-differentiable nature of indicator functions makes them unsuitable for use as barrier functions. Logarithmic functions satisfy the conditions stated above. Furthermore, their ability to grow exponentially almost mimics the jump seen in indicator functions.

Examples of barrier functions

- $X = \mathbb{R}^m$, $F(x) = -\sum_{i=1}^{m} \log(x_i)$
- $X = \mathcal{L}^m$, $F(x) = -\log(x_m^2 - \sum_{i=1}^{m-1} x_i^2)$
- $X = S_+^m$, $F(x) = -\log(\det(X))$
- For p inequality constraints of the type $f_i(x) \leq 0$, $F(x) = -\sum_{i=1}^{p} \log(f_i(x))$ can be used as the Barrier function.

The basic idea behind Barrier methods is as follows, we first rewrite the constrained convex optimization problem in 7.4 as follows,

$$\min \quad c^T x + \frac{1}{t} F(x) \tag{7.5}$$

where t is a parameter. If $t < \infty$ then the solution $x(t)$ to the above Problem (7.5) belongs to the interior of X, i.e. $x(t) \in \text{int}(X)$, and as t increases and $t \to \infty$, the solution to the above problem tends to the optimum, i.e. $x(t) \to x_*$. The Barrier method can be expressed in a pseudo-code see Algorithm 1.

Path Following Scheme

We can use the following equation to update t.

$$t_{k+1} = (1 + \frac{1}{13\sqrt{\nu}}) t_k \tag{7.6}$$

And use Newton's method in a sequential fashion to compute the minimum to Problem (7.5).

$$x_{k+1} = x_k - \left[\nabla^2 F(x_k)\right] \left[t_k c + \nabla F(x_k)\right] \tag{7.7}$$

Note that ν is a parameter associated with the Barrier function.

Algorithm 1 Barrier Method (Adapted from [BV04])

Input: Objective function $f(x)$, Barrier function $F(x)$, $x \in \text{int}(X)$,
    t Update Equation, Stopping Criteria (dependent on tolerance (ϵ))
Result: ϵ-suboptimal solution to Problem 7.4
repeat
    Centering Step: Compute $x_*(t) = \text{argmin}\{f(x) + (1/t)F(x)\}$ ▷ Use Eq. (7.7)
    Update: $x = x_*(t)$
    Increase t: Use t Update Equation ▷ Use Eq. (7.6)
until Stopping Criteria

Convergence

The error $\epsilon(x_k)$ is upper bounded by the following relation,

$$\epsilon(x_k) \triangleq \|c^T x_k - \min_{x \in X} c^T x\| \leq \frac{\left(2\nu e^{-\frac{k}{1+13\sqrt{\nu}}}\right)}{t_0}. \tag{7.8}$$

The number of steps or iterations required for the error function to drop below ϵ_0 is given by,

$$k \sim \mathcal{O}(\nu \log(\frac{\nu}{\epsilon_0})) \quad \text{for } \epsilon(x_k) \leq \epsilon_0 \tag{7.9}$$

The computational complexity of the barrier method is $\mathcal{O}(M\nu \log(1/\epsilon_0))$, where M is the cost of Newton step, usually of order $O(n^3)$. However the value of n can be quite large for large data sets. Algorithms dependent on n and higher powers of n significantly slow down with large datasets. Operations such as evaluating the Hessian is particularly expensive when n is large. It is hence critical to design algorithms with *dimension-free* complexity , if we wish to perform learning/inference/analytics over large data sets.

7.3 Introduction to Optimization Algorithms

A general optimization problem is defined as,

$$\begin{aligned} \min \quad & f(x) && \\ s.t. \quad & x \in X && \\ & g_i(x) \leq 0 && i = 1, \ldots, m \\ & h_j(x) = 0 && j = 1, \ldots, p \end{aligned} \tag{7.10}$$

where, $f(x)$ is the objective function, X is the feasible set, $g_i(x) \leq 0$ represents m inequality constraints and $h_j(x)$ represents p equality constraints. Recall from Definition 5.1, that convex optimization problems have linear equality constraints (or do not have equality constraints at all). In this section, we will look at a general taxonomy for optimization algorithms and few remarks on measures of error.

7.3.1 Black Box Oriented Algorithms

Black box oriented algorithms are algorithms that generate sequence of iterates $x_1, x_2, \ldots$, such that any new iterate x_{t+1} utilizes only *local information* collected along the search path (information at x_k or before). Unlike Interior Point Method, black box oriented algorithms usually do not exploit the structure of the problem.

Often analytic forms of the objective and/or constraint functions are unavailable and we may have to resort to using just function and gradient values at specified states [BBA]. Many optimization problems in finance (portfolio optimization), bioinformatics (protein folding), system theory (data drive system identification) etc. belong to this type. Black box algorithms are naturally suited for such problems. The optimizer queries the system to get function and gradient values at specific points (iterates) and then uses just this information (local) to perform optimization.

Depending on the information utilized, optimization algorithms can be classified into zero-order, first-order or second-order methods.

- **Zero-Order Methods** Optimization algorithms that utilize only function ($f(x_t)$) and constraint ($g_i(x_t)$) values are categorized as Zero-Order methods. Algorithms such as Nelder-Mead method, Random Search, Golden Section Search are Zero-Order methods. Several population based heuristic methods such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Simulated Annealing (SA) are also examples of Zero-Order methods. In population based methods, we start with a population of feasible candidate solutions, and then prune and improve the population using function values.
- **First-Order Methods** Optimization algorithms that utilize subgradients (or gradient) of the function ($\partial f(x_t)$) and constraint ($\partial g_i(x_t)$) along with function ($f(x_t)$) and constraint ($g_i(x_t)$) values are called First-Order Methods. Gradient Descent, Stochastic Gradient Descent, Coordinate Descent, Frank-Wolfe are some of the popular First-Order algorithms. First-order methods are extremely popular in large scale optimization, machine learning etc. due to low computational complexity and speed.
- **Second-Order Methods** Optimization algorithms that utilize hessian information ($\nabla^2 f(x_t)$, $\nabla^2 g_i(x_t)$), gradients information ($\nabla f(x_t)$, $\nabla g_i(x_t)$), and function and constraint ($f(x_t)$) ($g_i(x_t)$) values are called Second-Order Methods. Newton Method, BFGS method etc are some popular examples of the second-order methods.

Later in the course we will analyze algorithms to obtain theoretical guarantees on the correctness, performance and convergence speed of the optimization algorithms. Since large scale optimization problems (machine learning etc.) involve huge datasets with millions of model parameters to optimize for, it becomes extremely critical for the optimization algorithm to be computationally "simple". We will also analyze the trade-offs between convergence speed, computational complexity and optimality for optimization algorithms.

7.3.2 Measure of Error

A measure of error needs to be defined to evaluate the quality of solution obtained from any optimization algorithm (especially iterative algorithms). Let us denote error at solution x_t by $\epsilon(x_t)$. Note that we would want error to be non-negative ($\epsilon(x_t) \geq 0$) and it should approach zero as the solution approaches optima ($x_t \to x_* \implies \epsilon(x_t) \to 0$). We will list down a few metrics that satisfy the above conditions:

1. $\epsilon(x_t) = \inf_{x_* \in X_*} \|x_t - x_*\|$
   This metric represents the distance between the solution (x_t) and the optimal solution (x_*). It clearly satisfies both criteria mentioned above.
2. $\epsilon(x_t) = f(x_t) - f(x_*)$
   Here, $\epsilon(x_t)$ represents the mismatch between function values at a feasible solution ($f(x_t)$) and the optimum ($f(x_*)$). Clearly this metric possesses both the properties mentioned above. As $x_t \to x_*$, the function value $f(x_t) \to f(x_*)$ and $\epsilon(x_t) \to 0$.

3. $\epsilon(x_t) = \max(f(x_t) - f(x_*), [g_1(x_t)]_+, [g_2(x_t)]_+, \ldots, [g_m(x_t)]_+)$
   Note $[g_i(x_t)]_+ = \max(g_i(x_t), 0)$. In this metric, we try to bound both the optimality and infeasibility. This ensures that if $\epsilon(x_t) < \epsilon_0$ then both the function distance from optimum is less than ϵ_0 and the maximum constraint slackness is less than ϵ_0. That is, $\epsilon(x_t) < \epsilon_0 \implies f(x_t) < f(x_*) + \epsilon_0$ and $g_i(x_t) < \epsilon_0 \ \forall i = 1, \ldots, m$.

References

[NT08] A. Nemirovski and M. Todd, "Interior-point methods for optimization," *Acta Numerica*, Vol 17, 2008, pp. 191–234.

[K84] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, Vol 4, 1984, pp. 373-395.

[BV04] S. Boyd and L. Vandenberghe, "Convex Optimization," Cambridge University Press, 2004.

[BTN01] A. Ben-Tal and A. Nemirovski, "Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications," MPS-SIAM Series on Optimization,SIAM, 2001.

[DT06] G. B. Dantzig and M. N. Thapa, "Linear programming 2: theory and extensions," Springer Science & Business Media, 2006.

[DT06] D. G. Luenberger and Y. Ye, "Linear and Nonlinear Programming," Springer Science & Business Media, 2008.

[BBA] Black Box Algorithms, *http://archimedes.cheme.cmu.edu/?q=bbo* [Accessed: 09/13/16].

IE 598: Big Data Optimization **Fall 2016**

Lecture 8: Gradient Descent – September 15

Lecturer: Niao He *Scriber: Juho Kim*

Overview: In this lecture, we discussed the concept of the rate of convergence. We also started the discussion about smooth convex optimization. As the beginning of this, we studied the gradient descent method.

8.1 Recap

Recall that a typical form of an optimization problem is

$$\begin{aligned} \underset{x\in\mathcal{X}}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 0,\ i = 1, \dots, m. \end{aligned}$$

A black-box oriented algorithm generates $x_1, x_2, \dots$ in a way that x_{t+1} depends on local information gathered along $x_1, x_2, \dots, x_t$.

We can define a measure of error $\mathcal{E}(x_t)$ evaluate the quality of solution obtained from any optimization algorithm. The error function should be nonnegative, and it goes to zero as $x_t \to x_*$. For example, we can use the following measures as our error function.

1. $\mathcal{E}(x_t) = \inf_{x_* \in \mathcal{X}_*} \|x_t - x_*\|$
2. $\mathcal{E}(x_t) = \max\{f(x_t) - f(x_*),\ [g_1(x_t)]_+,\ \dots,\ [g_m(x_t)]_+\}$

8.2 Convergence Rate

Consider that $\lim_{t\to\infty} \frac{\mathcal{E}(x_{t+1})}{\mathcal{E}(x_t)^p} \le q$.

1. Linear convergence

   If $p = 1$ and $q \in (0, 1)$, the convergence rate is linear. For example, $\mathcal{E}(x_t) = O(e^{-at})\ (a > 0)$ converges linearly.
   In this case, the number of iterations required to obtain a solution within error ϵ is of order $O(\log \frac{1}{\epsilon})$.

2. Sublinear convergence

   If $p = 1$ and $q = 1$, the convergence rate is sublinear, which is slower than linear convergence. For example, $\mathcal{E}(x_t) = \frac{1}{t^b}\ (b > 0)$ converges sublinearly.
   In this case, the number of iterations required to obtain a solution within error ϵ is of order $O(\frac{1}{\epsilon^{1/b}})$.

3. Superlinear convergence

   If $p = 1$ and $q = 0$, the convergence rate is superlinear, which is faster than any geometric decay. For instance, $\mathcal{E}(x_t) = O(e^{-at^2})\ (a > 0)$ converges superlinearly.

4. Convergence of order p

   If $p = 2$ and $q > 0$, the convergence rate is quadratic (e.g., Newton's method). If $p = 3$ and $q > 0$, the convergence rate is cubic. $\mathcal{E}(x_t) = O(e^{-at^p})\,(a > 0)$ is the example of this convergence rate.
   In this case, the number of iterations required to obtain a solution within error ϵ is of order $O(\log(\log \frac{1}{\epsilon}))$.

For comparison of the rate of convergence, it is convenient to use $log(\mathcal{E}(x_t))$ instead of $\mathcal{E}(x_t)$ with respect to the number of iterations.

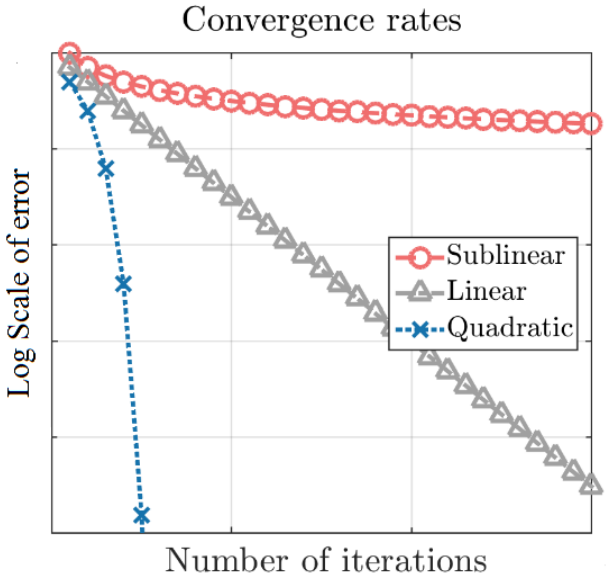


Figure 8.1: Comparison of the rate of convergence

The convergence rate of an optimization algorithm depends heavily on the structure of the problem (e.g. strong convexity, smoothness). We will first focus on smooth convex optimization.

8.3 Smooth Convex Optimization

Definition 8.1 ***(L-smooth)***
f is L-smooth (with a constant $L > 0$) on $\mathcal{X}$ if f is continuously differentiable and $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ for all $x, y \in \mathcal{X}$.

Proposition 8.2 *The followings are equivalent.*
(a) f is convex and L-smooth.
(b) $0 \leq f(y) - f(x) - \nabla f(x)^T(y - x) \leq \frac{L}{2}\|x - y\|_2^2$ for all $x, y \in \mathcal{X}$.
(c) $f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2$ for all $x, y \in \mathcal{X}$.
(d) $\{\nabla f(x) - \nabla f(y)\}^T(x - y) \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2$ for all $x, y \in \mathcal{X}$.

Proof: (a) ⇒ (b) By the fundamental theorem of calculus,

$$\begin{aligned}
f(y) - f(x) - \nabla f(x)^T(y-x) &= \int_0^1 \nabla f(x + t(y-x))^T(y-x)\,dt - \nabla f(x)^T(y-x) \\
&= \int_0^1 [\nabla f(x + t(y-x)) - \nabla f(x)]^T(y-x)\,dt \\
&\le \int_0^1 \|\nabla f(x + t(y-x)) - \nabla f(x)\|_2 \|y-x\|_2\,dt \quad \text{(by Cauchy-Schwarz inequality)} \\
&\le \int_0^1 L\|t(y-x)\|_2\|y-x\|_2\,dt \quad \text{(by L-smoothness of } f\text{)} \\
&= L\|(y-x)\|_2^2 \int_0^1 t\,dt \\
&= \frac{L}{2}\|(y-x)\|_2^2
\end{aligned}$$

(b) ⇒ (c) Let $z = y + \frac{1}{L}(\nabla f(x) - \nabla f(y))$

$$\begin{aligned}
f(y) - f(x) &= f(y) - f(z) + f(z) - f(x) \\
&\ge -\nabla f(y)^T(z-y) - \frac{L}{2}\|y-z\|_2^2 + \nabla f(x)^T(z-x) \\
&= \nabla f(x)^T(y-x) - \{\nabla f(x) - \nabla f(y)\}^T(y-z) - \frac{L}{2}\|y-z\|_2^2 \quad \text{(by plugging in z)} \\
&= \nabla f(x)^T(y-x) + \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2 - \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|_2^2 \\
&= \nabla f(x)^T(y-x) + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2
\end{aligned}$$

(c) ⇒ (d) Suppose that

$$\begin{aligned}
f(y) &\ge f(x) + \nabla f(x)^T(y-x) + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2 \\
f(x) &\ge f(y) + \nabla f(y)^T(x-y) + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|_2^2
\end{aligned}$$

By summing up two inequalities, we can obtain

$$[\nabla f(x) - \nabla f(y)]^T(x-y) \ge \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

(d) ⇒ (a) Suppose that $[\nabla f(x) - \nabla f(y)]^T(x-y) \ge \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2$.

By Cauchy-Schwarz inequality,

$$\|\nabla f(x) - \nabla f(y)\|_2\|x-y\|_2 \ge \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|_2^2$$

which implies f is L-smooth. The convexity of f is due to the following claim.
Therefore, (a), (b), (c), and (d) are equivalent. ■

Claim 8.3 *Suppose f is differentiable, then f is convex if and only if $[\nabla f(x) - \nabla f(y)]^T(x-y) \ge 0$.*

Proof:
($\Rightarrow$) Suppose f is convex. Then,

$$f(x) \geq f(y) + \nabla f(y)^T(x-y) \quad \textit{and} \quad f(y) \geq f(x) + \nabla f(x)^T(y-x)$$

which imply $[\nabla f(x) - \nabla f(y)]^T(x-y) \geq 0$.

($\Leftarrow$) Suppose $[\nabla f(x) - \nabla f(y)]^T(x-y) \geq 0$.

$$f(y) - f(x) - \nabla f(x)^T(y-x) = \int_0^1 \frac{1}{t}\{\nabla f(x+t(y-x)) - \nabla f(x)\}^T t(y-x)\, dt \geq 0$$

which implies f is convex. ■

Remark. In general, we call an operator $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ *monotonic* if $\langle F(x) - F(y),\, x-y\rangle \geq 0$. The above claim indicates that the gradient of a continuously differentiable convex function is an monotone operator.

8.4 Gradient Descent

Consider an unconstrained optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)$$

where f is L-smooth and convex.

Gradient Descent (GD) Given a starting point $x_0 \in \operatorname{dom} f$, and step-size $\gamma > 0$, GD works as follows,

$$x_{t+1} = x_t - \gamma \nabla f(x_t), \quad t = 0, 1, 2, \ldots$$

until stopping criterion is satisfied.

Observation 1 The gradient descent step can be regarded as minimizing a quadratic approximation of the objective function,

$$\begin{aligned} x_{t+1} &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \; \{f(x_t) + \nabla f(x_t)^T(x-x_t) + \frac{1}{2\gamma}\|x-x_t\|_2^2\} \\ &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \; \left\{\frac{1}{2\gamma}\|x-(x_t-\gamma\nabla f(x_t)\|_2^2\right\} \end{aligned}$$

When $\gamma \leq \frac{1}{L}$, the quadratic approximation is indeed an upper bound of the objective function.

Observation 2 The gradient descent step strictly reduces the objective function value at each iteration,

$$f(x_{t+1}) - f(x_t) \leq \nabla f(x_t)^T(-\gamma\nabla f(x_t)) + \frac{L}{2}\|-\gamma\nabla f(x_t)\|_2^2 = -\gamma(1-\frac{L}{2}\gamma)\|\nabla f(x_t)\|_2^2$$

It makes sense to seek $\gamma > 0$ that maximizes $\gamma(1-\frac{L}{2}\gamma)$ to ensure the most reduction. This takes place when $\gamma = \frac{1}{L}$ and moreover,

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L}\|\nabla f(x_t)\|_2^2.$$

Theorem 8.4 *Let f be convex and L-smooth, x_* be an optimal solution. With $\gamma = \frac{1}{L}$ for all $\gamma > 0$, the gradient descent satisfies*

$$f(x_t) - \min_{x \in \mathbb{R}^n} f(x) \leq \frac{2L\|x_0 - x_*\|_2^2}{t}$$

Proof: ([Nes03]) First of all, we have the following

(i) $f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L}\|\nabla f(x_t)\|_2^2$ from Observation 2.

(ii) $\|x_{t+1} - x_*\|_2 \leq \|x_t - x_*\|_2$
This is because

$$\begin{aligned}
\|x_{t+1} - x_*\|_2^2 &= \|x_t - \frac{1}{L}\nabla f(x_t) - x_*\|_2^2 \\
&= \|x_t - x_*\|_2^2 - \frac{2}{L}\nabla f(x_*)^T(x_t - x_*) + \frac{1}{L^2}\|\nabla f(x_t)\|_2^2 \\
&\leq \|x_t - x_*\|_2^2 - \frac{1}{L^2}\|\nabla f(x_t)\|_2^2 \quad \text{since } \nabla f(x_*)^T(x_t - x_*) \geq \frac{1}{L}\|\nabla f(x_t)\|_2^2 \\
&\leq \|x_t - x_*\|_2^2
\end{aligned}$$

(iii) $\|\nabla f(x_t)\|_2 \geq \frac{f(x_t) - f(x_*)}{\|x_t - x_*\|}$
since $f(x_t) - f(x_*) \leq \nabla f(x_t)^T(x_t - x_*) \leq \|\nabla f(x_t)\|_2\|x_t - x_*\|_2$.

By combining (i)-(iii), we arrive at

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L}\Big[\frac{f(x_t) - f(x_*)}{\|x_0 - x_*\|_2}\Big]^2.$$

Let $\epsilon_t = f(x_t) - f(x_*)$ and $\beta = \frac{1}{2L\|x_0 - x_*\|_2^2}$.

$$[f(x_{t+1}) - f(x_*)] - [f(x_t) - f(x_*)] = \epsilon_{t+1} - \epsilon_t \leq -\frac{1}{2L}\frac{\epsilon_t^2}{\|x_0 - x_*\|_2^2} = -\beta\epsilon_t^2$$

$$\frac{1}{\epsilon_t} - \frac{1}{\epsilon_{t+1}} \leq -\beta\frac{\epsilon_t}{\epsilon_{t+1}} \leq -\beta$$

$$\begin{aligned}
&\Rightarrow \ \frac{1}{\epsilon_t} + \beta \leq \frac{1}{\epsilon_{t+1}} \\
&\Rightarrow \ \frac{1}{\epsilon_0} + \beta t \leq \frac{1}{\epsilon_t} \\
&\Rightarrow \ \beta t \leq \frac{1}{\epsilon_t} = \frac{1}{f(x_t) - f(x_*)}
\end{aligned}$$

which implies $f(x_t) - f(x_*) \leq \frac{2L\|x_0 - x_*\|_2^2}{t}$. ■

References

[Nes03] Y. Nesterov, "Introductory Lectures on Convex Optimization, A Basic Course," *Springer*, 2003.

[BV04] S. Boyd and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, 2004.

IE 598: Big Data Optimization **Fall 2016**

Lecture 9: Gradient Descent and Acceleration – September 20

Lecturer: Niao He *Scriber: Samantha Thrush*

In the previous lecture, we had introduced Gradient Descent for Smooth Convex Optimization problems. In this lecture, we will delve further in to the details of Gradient Descent for strongly convex problems and will discuss the mathematical and historical background of Accelerated Gradient Descent.

9.1 Review

Recall from last time that we considered the following unconstrained problem:

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

where f is L-smooth and convex.

The simplest gradient descent acts as follows:

$$x_{t+1} = x_t - \frac{1}{L}\nabla f(x_t)$$

which enjoys a $O(1/t)$ sublinear rate of convergence,

$$f(x_t) - f(x_*) \leq \frac{2L||x_0 - x_*||^2}{t}$$

The key to this convergence is based on the following observation:

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L}||\nabla f(x_t)||_2^2$$

Now we want to take this one step further and show that Gradient Descent achieves a linear convergence rate for strongly convex problems.

9.2 Strongly Convex Problems

Definition 9.1 *f is μ-strongly convex ($\mu \geq 0$) if $f(x) - \frac{\mu}{2}||x||_2^2$ is convex, where μ is the strong-convexity parameter.*

Proposition 9.2 *The following are all equivalent:*

(a) *f is continuously differentiable and μ-strongly convex*

(b) *f is continuously differentiable and $f(\alpha x+(1-\alpha)y) \leq \alpha f(x)+(1-\alpha)f(y)-\frac{\mu}{2}\alpha(1-\alpha)||x-y||_2^2, \forall \alpha \in [0,1]$*

(c) *$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}||x-y||_2^2$*

(d) *$\langle \nabla f(x) - \nabla f(y), x-y\rangle \geq \mu||x-y||_2^2$*

Remark 1. Recall the previous notes on equivalent characterization for L-smooth functions, one could observe that there exist some laws of symmetry between L-smoothness and μ-convexity. Notice that if $\mu = 0$ then properties (b), (c), (d) will reduce to a form that is applicable to general convex functions. If a function f is both L-smooth and μ-strongly convex, then usually we have $\mu \leq L$.

Remark 2. If f is further twice-differentiable, then f is μ-strongly convex if and only if $\nabla^2 f \succeq \mu I$.

Proof:

- **(a)** $\iff$ **(b)** Since $f(x) - \frac{\mu}{2}\|x\|_2^2$ is convex, this implies that for any $x, y, \alpha \in [0, 1]$,

  $$f(\alpha x + (1-\alpha)y) - \frac{\mu}{2}\|\alpha x + (1-\alpha)y\|_2^2 \leq \alpha[f(x) - \frac{\mu}{2}\|x\|_2^2] + (1-\alpha)[f(y) - \frac{\mu}{2}\|y\|_2^2]$$

  Rearranging the terms, we will arrive at (b).

- **(b)** $\implies$ **(c)** From (b), we have $(1-\alpha)f(y) - (1-\alpha)f(x) \geq f(\alpha x + (1-\alpha)y) - f(x) + \frac{\mu}{2}\alpha(1-\alpha)\|x-y\|_2^2$. Hence,

  $$f(y) - f(x) \geq \frac{f(x + (1-\alpha)(y-x)) - f(x)}{1-\alpha} + \frac{\mu}{2}\alpha\|x-y\|_2^2, \forall \alpha \in [0, 1]$$

  Let α goes to one, thus leading to (c).

- **(c)** $\implies$ **(d)** From (c), we have

  $$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}||x-y||_2^2$$

  $$f(x) \geq f(y) - \nabla f(y)^T(x-y) + \frac{\mu}{2}||x-y||_2^2$$

  Adding these two equations together leads to (d).

- **(d)** $\implies$ **(b)** Let $z = \alpha x + (1-\alpha)y$, from the fundamental theory of calculus, we can see the following:

  $$f(z) = f(x) + \textstyle\int_0^1 \nabla f(x + t(z-x))^T(z-x)dt = f(x) + (1-\alpha)\displaystyle\int_0^1 \nabla f(x + t(z-x))^T(y-x)dt$$

  $$f(z) = f(y) + \textstyle\int_0^1 \nabla f(y + t(z-y))^T(z-y)dt = f(y) - \alpha\displaystyle\int_0^1 \nabla f(x + t(z-x))^T(y-x)dt$$

  If we multiply this first integral equation by α and the second by $(1-\alpha)$ and add them together, we end up with

  $$\alpha f(x) + (1-\alpha)f(y) - f(z) = \alpha(1-\alpha)\int_0^1 [\nabla f(x + t(z-x)) - \nabla f(y + t(z-y))](x-y)dt$$

  From equation (b) above, we further have

  $$\alpha f(x) + (1-\alpha)f(y) - f(z) \geq \mu\alpha(1-\alpha)||x-y||^2 \int_0^1 (1-t)dt = \frac{1}{2}\mu\alpha(1-\alpha)||x-y||^2$$

  Hence, we have shown that f is μ-strongly convex.

■

Proposition 9.3 *Assume f is μ-strongly convex and differentiable. Let $x_* \in argmin_{x\in\mathbb{R}^n} f(x)$, the following must then be true for any $x \in \mathbb{R}^n$:*

(i) $f(x) - f(x_*) \geq \frac{\mu}{2}||x - x_*||_2^2$

(ii) $f(x) - f(x_*) \leq \frac{1}{2\mu}||\nabla f(x)||_2^2$

Proof:

(i) Following from equation (c) of Proposition 9.2, we have

$$f(x) \geq f(x_*) + \nabla f(x_*)^T(x - x*) + \frac{\mu}{2}||x - x_*||_2^2$$

Since x_* is the minimizer of $f(x)$, this means that $\nabla f(x_*) = 0$. Hence,

$$f(x) \geq f(x_*) + \frac{\mu}{2}||x - x_*||_2^2$$

(ii) First, we start with equation (c) from Proposition 9.2.

$$f(x_*) \geq f(x) + \nabla f(x)^T(x_* - x) + \frac{\mu}{2}||x - x_*||_2^2$$

We can then re-arrange this equation so that it has a similar form to equation (ii) in Proposition 9.3

$$f(x) - f(x_*) \leq \nabla f(x)^T(x - x_*) - \frac{\mu}{2}||x - x_*||_2^2$$

Therefore,

$$f(x) - f(x_*) \leq ||\nabla f(x)||_2 \cdot ||x - x_*||_2 - \frac{\mu}{2}||x - x_*||_2^2 \leq \max_{\alpha>0} ||\nabla f(x)||_2 \cdot \alpha - \frac{\mu}{2}\alpha^2 = \frac{1}{2\mu}||\nabla f(x)||_2^2$$

Thus, proving equation (ii) in Proposition 9.3.

■

9.3 Gradient Descent for Strongly Convex Problems

In the following, we consider unconstrained optimization problem

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

where f is L-smooth and μ-strongly convex. We try to analyze the convergence of the gradient descent

$$x_{t+1} = x_t - \gamma\nabla f(x_t), t = 0, 1, \ldots$$

under two different choices of stepsize.

9.3.1 Stepsize $\gamma = 1/L$

Theorem 9.4 *Let f be L-smooth and μ-strongly convex, with step size $\gamma = \frac{1}{L}$, Gradient Descent satisfies*

$$f(x_t) - f(x_*) \leq (1 - \frac{\mu}{L})^t [f(x_0) - f(x_*)]$$
$$||x_t - x_*||_2^2 \leq \frac{2}{\mu}(1 - \frac{\mu}{L})^t [f(x_0) - f(x_*)]$$

Remark: This implies a linear convergence of the Gradient Descent method for solving strongly convex problems. The term $\frac{L}{\mu}$, usually denoted as κ, is known as the condition number. The smaller the condition number is, the faster the convergence will be.

Proof: We know that the following is true due to properties of L-smoothness (which is responsible for the right-most term) and μ-strong convexity (which gives the left-most term):

$$2\mu[f(x_t) - f_*] \leq ||\nabla f(x_t)||_2^2 \leq 2L[f(x_t) - f(x_{t+1})]$$

Let $\epsilon_t = f(x_t) - f(x_*)$, then we have $2\mu\epsilon_t \leq 2L \cdot (\epsilon_t - \epsilon_{t+1})$.

Re-arranging this equation leads to

$$\epsilon_{t+1} \leq (1 - \frac{\mu}{L})\epsilon_t$$

Hence, by induction, we have

$$\epsilon_t \leq (1 - \frac{\mu}{L})^t \epsilon_0, \forall t \geq 1.$$

Invoking the fact that $f(x) - f(x_*) \geq \frac{\mu}{2}\|x - x_*\|_2^2$, we further have

$$||x_t - x_*||_2^2 \leq \frac{2}{\mu}(1 - \frac{\mu}{L})^t [f(x_0) - f(x_*)].$$

■

9.3.2 Stepsize $\gamma = 2/(\mu + L)$

Proposition 9.5 *If f is μ-strongly convex and L-smooth, then the following is true:*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu}{\mu + L}||x - y||^2 + \frac{1}{\mu + L}||\nabla f(x) - \nabla f(y)||^2$$

Proof: Let us first consider the case when $\mu = L$, from the μ-strongly convexity and L-smoothness of f, we have respectively,

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \mu||x - y||_2^2$$
$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{1}{L}||\nabla f(x) - \nabla f(y)||_2^2$$

Adding these two equations together leads to

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle > \frac{\mu}{2}||x - y||^2 + \frac{1}{2L}||\nabla f(x) - \nabla f(y)||^2$$

as expected.

Now, let us explore the case of $\mu < L$. Define the function $\phi(x) = f(x) - \frac{\mu}{2}||x||_2^2$. So $\phi(x)$ is convex and $(L-\mu)$-smooth, which is equivalent to

$$\langle \nabla\phi(x) - \nabla\phi(y), x - y\rangle \geq \frac{1}{L-\mu}||\nabla\phi(x) - \nabla\phi(y)||_2^2$$

Plugging in the definition of $\phi(x)$ gives

$$\langle [\nabla f(x) - \nabla f(y)] - \mu[x-y], x - y\rangle \geq \frac{1}{L-\mu}||[\nabla f(x) - \nabla f(y)] - \mu[x-y]||_2^2$$

Expanding the norms and rearranging the terms will then lead to the conclusion.

■

Theorem 9.6 *Let f be L-smooth and μ-strongly convex, with step size $\gamma = \frac{2}{\mu+L}$, Gradient Descent satisfies*

$$f(x_t) - f(x_*) \leq \frac{L}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^{2t}||x_0 - x_*||^2$$
$$||x_t - x_*||_2^2 \leq \left(\frac{\kappa-1}{\kappa+1}\right)^{2t}||x_0 - x_*||^2$$

As stated before, $\kappa = \frac{L}{\mu}$.

Proof: The proof of the above proposition is straightforward:

$$\begin{aligned}
||x_{t+1} - x_*||_2^2 &= ||x_t - \frac{2}{\mu+L}\nabla f(x_t) - x_*||_2^2 \\
&= ||x_t - x_*||^2 - \frac{4}{\mu+L}\langle \nabla f(x_t), x_t - x_*\rangle + \frac{4||\nabla f(x_t)||_2^2}{(\mu+L)^2} \\
&\leq ||x_t - x_*||^2 - \frac{4}{\mu+L}\left[\frac{\mu L}{\mu+L}||x_t - x_*||^2 + \frac{1}{\mu+L}||\nabla f(x_t)||_2^2\right] + \frac{4||\nabla f(x_t)||_2^2}{(\mu+L)^2} \\
&\leq \left[1 - \frac{4\mu L}{(\mu+L)^2}\right]||x_t - x_*||_2^2 \\
&\leq \left(\frac{\mu-L}{\mu+L}\right)^2||x_t - x_*||_2^2 \\
&= \left(\frac{\kappa-1}{\kappa+1}\right)^2||x_t - x_*||_2^2
\end{aligned}$$

By induction, we have

$$||x_t - x_*||_2^2 \leq \left(\frac{\kappa-1}{\kappa+1}\right)^{2t}||x_0 - x_*||^2.$$

Furthermore, by L-smoothness, we have $f(x_t) - f(x_*) \leq \frac{L}{2}\|x_t - x_*\|_2^2$, therefore,

$$f(x_t) - f(x_*) \leq \frac{L}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^{2t}||x_0 - x_*||^2.$$

■

Discussion. In the subsections above, we have seen that the Gradient Descent exhibit slightly different linear convergences under two different choices of step-sizes:

Before: $\gamma = \frac{1}{L}$ which gave a rate of order $\left(1 - \frac{\mu}{L}\right)^t = \left(1 - \frac{1}{\kappa}\right)^t \leq e^{-t/\kappa}$.

Now: $\gamma = \frac{2}{\mu+L}$ which gives a rate of order $\left(1 - \frac{2}{\kappa+1}\right)^{2t} \leq e^{-4t/(\kappa+1)}$, which is better than $e^{-t/\kappa}$.

9.4 Accelerated Gradient Descent

9.4.1 Overview

Question: is Gradient Descent the best algorithm to solve smooth convex optimization problems? Is it possible to obtain a even faster rate of convergence?
It was shown in Nesterov's seminal work in 1983 that the answer is yes, and a better (indeed optimal) convergence rate can be achieved through Accelerated Gradient Descent algorithm. Here is a brief summary.

Problem type	GD Convergence Rate	AGD Convergence Rate
f is L-smooth and convex	$\mathcal{O}\left(\frac{LD^2}{t}\right)$	$\mathcal{O}\left(\frac{LD^2}{t^2}\right)$
f is L-smooth and μ-strongly convex	$\mathcal{O}\left(\left(\frac{\kappa-1}{\kappa+1}\right)^{2t}\right)$	$\mathcal{O}\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2t}\right)$

9.4.2 Historical Note of Accelerated Gradient Descent

The following will detail the history and evolution of the study of accelerated gradient descent:

- [YN83] In 1983, Nesterov created the first accelerated gradient descent scheme for smooth functions.
- [YN88] In 1988, Nesterov published another, more general acceleration scheme for smooth functions.
- [YN04] In 2004, Nesterov combined the acceleration scheme with smoothing techniques to address non-smooth problems.
- [YN07] In 2007, Nesterov further extended the accelerated gradient descent to solve composite problems (problems that have both smooth and non-smooth parts).
- [BT08] In 2008, Beck and Teboulle established another simple and popular acceleration algorithm, FISTA (Fast Iterative Shrinkage-Thresholding Algorithm), designed to solve composite problems.
- [TY09] In 2009, Tseng performed unified analysis of both smooth and non-smooth problems under accelerated gradient descent.
- Since then, this leads to an interesting "acceleration phenomenon" and the acceleration idea has been applied to many other first-order algorithms,

Despite of its significance, the mechanism behind this algorithm has long been conceived as pure "algebraic trick" and remains mysterious. Many interesting interpretations have been recently developed from different viewpoints, see more details from the following references.

- **ODE perspective:**

  - [SB15] Su, Boyd, and Candes, 2015."A differential equation for modeling Nesterovs accelerated gradient method: theory and insights."
  - [LR14] Lessard, Racht, Packward, 2014."Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints."
  - [WW16] Wibisono, Wilson, and Jordan, 2016. "A Variational Perspective on Accelerated Methods in Optimization."

- **Geometry perspective:**
  - [AZ14] Allen-Zhu and Orrchia, 2014. "Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent."
  - [BL15] Bubeck, Lee and Singh, 2015. "A geometric alternative to Nesterovs accelerated gradient descent."

- **Game theory perspective:**
  - [LZ15] Lan and Zhou, 2015. "An optimal randomized incremental gradient method."

9.4.3 Accelerated Gradient Descent

We will look at one of Nesterov's Optimal Gradient Scheme for the unconstrained smooth convex optimization

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

where f is L-smooth and μ-strongly convex ($\mu \geq 0$).

The Accelerated Gradient Descent works as follows: let $y_0 = x_0$, for $t = 0, 1, 2, ...$, update

$$x_{t+1} = y_t - \frac{1}{L}\nabla f(y_t)$$
$$y_{t+1} = x_{t+1} + \beta_t(x_{t+1} - x_t), \text{with } \beta_t = \frac{\alpha_t(1-\alpha_t)}{\alpha_t^2 + \alpha_{t+1}}$$

where α_{t+1}^2 is defined in the following way:

$$\alpha_{t+1}^2 = (1-\alpha_{t+1})\alpha_t^2 + \frac{\mu}{L}\alpha_{t+1}$$

The above scheme is indeed very general and flexible. We list a few simple variants below under particular choices of parameters.

- **Heavy-ball method:** This method is named the heavy-ball method as it has a momentum term in the x_{t+1} term ($\beta_t(x_t - x_{t-1})$), which allows for the equation to imitate the motion of a heavy ball rolling down a hill.
$$x_{t+1} = x_t - \alpha_t\nabla f(x_t) + \beta_t(x_t - x_{t-1})$$
where α_t and β_t are set to $\alpha_t = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \beta_t = \frac{(\sqrt{L}-\sqrt{\mu})^2}{(\sqrt{L}+\sqrt{\mu})^2}$.

- **Nesterov'83:-**
$$\begin{cases} x_{t+1} = y_t - \frac{1}{L}\nabla f(y_t) \\ y_{t+1} = x_{t+1} + \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}(x_{t+1} - x_t) \end{cases}$$

- **FISTA:**

$$\begin{cases} x_{t+1} = y_t - \frac{1}{L}\nabla f(y_t) \\ y_{t+1} = x_{t+1} + \frac{\lambda_t - 1}{\lambda_{t+1}}(x_{t+1} - x_t) \end{cases}$$

  where λ is defined in the following way:

$$\lambda_0 = 0, \lambda_{t+1} = \frac{1 + \sqrt{1 + 4\lambda_t^2}}{2}, \forall t \geq 0$$

- **Nesterov'88:**

$$\begin{cases} x_{t+1} = y_t - \frac{1}{L}\nabla f(y_t) \\ y_{t+1} = x_{t+1} + \frac{t}{t+3}(x_{t+1} - x_t) \end{cases}$$

The first two variants are widely used for strongly convex problems and the latter two are used for general smooth convex problems. In the following, we are going to analyze the convergence rate for the FISTA algorithm.

Theorem 9.7 *Let f be L-smooth and convex, then the solution x_t from the FISTA algorithm satisfies:*

$$f(x_t) - f(x_*) \leq \frac{2L \cdot ||x_0 - x_*||_2^2}{t^2}$$

Proof: We start with the following fact:

$$f(x_{t+1}) - f(x) = f(x_{t+1}) - f(y_t) + f(y_t) - f(x)$$

Using the facts that $f(x_{t+1}) - f(y_t) \leq -\frac{1}{2L}\|\nabla f(y_t)\|_2^2$ and $f(y_t) - f(x) \geq \nabla f(y_t)^T(y_t - x)$, we have

$$f(x_{t+1}) - f(x) \leq -\frac{1}{2L}||\nabla f(y_t)||_2^2 + \nabla f(y_t)^T(y_t - x)$$

Hence, by definition of $x_{t+1} = y_t - \frac{1}{L}\nabla f(y_t)$, we have

$$f(x_{t+1}) - f(x) \leq -\frac{L}{2}||x_{t+1} - y_t||_2^2 - L(x_{t+1} - y_t)^T(y_t - x)$$

Let $x = x_t$ and $x = x_*$, respectively, and multiply the resulting equation with two separate factors:

$$[f(x_{t+1}) - f(x_t)](\lambda_t - 1) \leq \left[-\frac{L}{2}||x_{t+1} - y_t||_2^2 - L(x_{t+1} - y_t)^T(y_t - x_t)\right](\lambda_t - 1)$$

$$[f(x_{t+1}) - f(x_*)] \leq \left[-\frac{L}{2}||x_{t+1} - y_t||_2^2 - L(x_{t+1} - y_t)^T(y_t - x_*)\right]$$

Note that $\lambda_t \geq 1, \forall t \geq 1$. Adding these two equations together, we arrive at

$$\lambda_t f(x_{t+1}) - (\lambda_t - 1)f(x_t) - f(x_*) \leq -\frac{\lambda_t L}{2}||x_{t+1} - y_t||_2^2 - L(x_{t+1} - y_t)^T(\lambda_t y_t + (\lambda_t - 1)x_t - x_*)$$

Let $\epsilon_t = f(x_t) - f(x_*)$, this leads to :

$$\lambda_t \epsilon_{t+1} - (\lambda_t - 1)\epsilon_t \leq -\frac{\lambda_t L}{2}||x_{t+1} - y_t||_2^2 - L(x_{t+1} - y_t)^T(\lambda_t y_t + (\lambda_t - 1)x_t - x_*)$$

Multiplying both sides by λ_t:

$$\lambda_t^2\epsilon_{t+1} - \lambda_t(\lambda_t - 1)\epsilon_t \leq -\frac{L}{2}\left[\lambda_t^2||x_{t+1} - y_t||_2^2 + 2\lambda_t L(x_{t+1} - y_t)^T(\lambda_t y_t + (\lambda_t - 1)x_t - x_*)\right]$$

By definition of λ_t, we know the following is true,

$$\lambda_{t+1}^2 - \lambda_{t+1} = \lambda_t^2$$

Rearranging terms, we have

$$\lambda_t^2\epsilon_{t+1} - \lambda_{t-1}^2\epsilon_t \leq -\frac{L}{2}(||\lambda_t x_{t+1} - (\lambda_t - 1)x_t - x_*||^2 - ||\lambda_t y_t - (\lambda_t - 1)x_t - x_*||^2)$$

Invoking the definition of y_{t+1}, one can show that

$$\lambda_t x_{t+1} - (\lambda_t - 1)x_t - x_* = \lambda_{t+1}y_{t+1} - (\lambda_{t+1} - 1)x_{t+1} - x_*$$

Hence, defining $u_0 = x_0, u_t = \lambda_t y_t - (\lambda_t - 1)x_t - x_*, \forall t \geq 1$, the above equation simplifies to

$$\lambda_t^2\epsilon_{t+1} - \lambda_{t-1}^2\epsilon_t \leq -\frac{L}{2}(||u_{t+1}||^2 - ||u_t||^2)$$

Therefore, by induction, we have

$$\lambda_{t-1}^2\epsilon_t \leq \frac{L}{2}||u_0||^2$$

From this, we realize the following:

$$\epsilon_t \leq \frac{L||x_0||^2}{2\lambda_{t-1}^2}$$

By simple induction, we can show that

$$\lambda_{t-1} \geq \frac{t}{2}, \forall t \geq 1$$

Therefore:

$$\epsilon_t \leq \frac{2L||x_0||^2}{t^2}$$

■

References

[AZ14] Z. Allen-Zhu and L. Orecchia, "Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent", *ArXiv*, 2014, arXiv:1407.1537v4.

[BT08] A. Beck, and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", *SIAM journal on imaging sciences*, 2008, vol. 2, num1, pp. 183-202.

[BL15] S. Bubeck, Y. T. Lee, and M. Singh, "A geometric alternative to Nesterov's accelerated gradient descent", *ArXiv*, 2015, arXiv:1506:08187v1.

[LZ15] G. Lan and Y. Zhou, "An optimal randomized incremental gradient method", *ArXiv*, 2015, arXiv:1507.02000v3.

[LR14] L. Lessard, B. Recht, and A. Packard, "Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints", *Arxiv*, 2014, arXiv: 1408.3595v7.

[YN83] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O (1/k2)," *Soviet Mathematics Doklady*, 1983, vol. 27 num. 2, pp. 372–376.

[YN88] Y. Nesterov and A. Nemirovsky "A general approach to polynomial-time algorithms design for convex programming," *Report, Central Economical and Mathematical Institute, USSR Academy of Sciences, Moscow*, 1988.

[YNBK] Y. Nesterov, "Introductory Lectures on Convex Optimization: A Basic Course", *Kluwer-Academic*, 2003.

[YN04] Y. Nesterov, "Introductory Lectures on Convex Optimization. Applied Optimization" , *Kluwer Academic Publishers, Boston*, 2004, vol. 87.

[YN07] Y. Nesterov, "Gradient methods for minimizing composite objective function", *UCL*, 2007.

[SB15] W. Su, S. Boyd, and E.J. Candes , "A differential equation for modeling Nesterovs accelerated gradient method: theory and insights", *ArXiv*, 2015, arXiv:1503.01243.

[TY09] P. Tseng and S. Yun "A coordinate gradient descent method for nonsmooth separable minimization", *Mathematical Programming*, 2009, vol. 117, pp. 387-423.

[WW16] A. Wibisono, A. Wilson, and M. Jordan, "A Variational Perspective on Accelerated Methods in Optimization", *ArXiv*, 2016, arXiv:1603.04245.

IE 598: Big Data Optimization **Fall 2016**

Lecture 10: Lower bounds & Projected Gradient Descent– September 22

Lecturer: Niao He *Scriber: Meghana Bande*

Overview: In this lecture, we discuss the lower bounds on the complexity of first order optimization algorithms. We introduce the Projected Gradient Descent for constrained optimization problems and discuss their convergence rates. We illustrate how to find the projections for a few convex sets using examples.

10.1 Recap

In the previous lecture, we have seen the convergence rates of gradient descent and accelerated gradient descent algorithms which can be summarized below. Note that $\kappa = \frac{L}{\mu}$.

	Gradient Descent	Accelerated Gradient Descent
f is convex, L smooth	$O(\frac{LD^2}{t})$	$O(\frac{LD^2}{t^2})$
f is μ strongly convex, L smooth	$O((\frac{\kappa-1}{\kappa+1})^{2t})$	$O((\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^{2t})$

In order to show the optimality of these algorithms in terms of complexity, lower bounds will be discussed in this lecture.

10.2 Lower Bounds

We look at lower complexity bounds for convex optimization problems which use first order methods for objective functions belonging to certain classes .

10.2.1 Lower Bound for Smooth Convex Problems

Theorem 10.1 *For any t, $1 \leq t \leq \frac{1}{2}(n-1)$, and any $x_0 \in \mathbb{R}^n$, there exists an L-smooth convex function f such that, for any first order method $\mathcal{M}$ that generates a sequence of points x_t such that $x_t \in x_0 + \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_{t-1})\}$, we have*

$$f(x_t) - f(x^*) \geq \frac{3L\|x_0 - x^*\|^2}{32(t+1)^2}.$$

Proof: The sequence of iterates for $f(x)$ starting from x_0 is just a shift of the sequence generated for $f(x+x_0)$ from the origin. Without loss of generality, we assume that $x_0 = 0$.

Consider function f where

$$f(x) = \frac{L}{4} \cdot (\frac{1}{2}x^T A x - e_1^T x),$$

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & \ddots \\ 0 & 0 & \ddots & \ddots \end{pmatrix}_{(n\times n)} \quad \text{and } e_1 = \begin{pmatrix} 1 \\ 0 \\ . \\ 0 \end{pmatrix}_{n\times 1}.$$

1. We show that the function f is convex and L smooth. We have $\nabla^2 f(x) = \frac{L}{4}A$ and we need to show that $0 \preceq A \preceq 4I$. For any $x \in \mathbb{R}^n$, we have

$$\begin{aligned} x^T Ax &= 2\sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n-1} x_i x_{i+1} \\ &= x_1^2 + x_n^2 + \sum_{i=1}^{n-1}(x_i - x_{i+1})^2. \end{aligned}$$

   From this, we have $0 \preceq A$. In order to show $A \preceq 4I$ consider the following,

$$\begin{aligned} x^T Ax &= x_1^2 + x_n^2 + \sum_{i=1}^{n-1}(x_i - x_{i+1})^2 \\ &\leq x_1^2 + x_n^2 + 2\sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2) \\ &\leq 4\sum_{i=1}^{n} x_i^2. \end{aligned}$$

   where we have used $(a-b)^2 \leq 2(a^2+b^2)$ since $(a-b)^2 + (a+b)^2 = 2(a^2+b^2)$.

2. We now find the optimal value and optimal solution of the function f. To this end, we set $\nabla f(x) = 0$ which gives us $Ax - e_1 = 0$ or the set of equations

$$2x_1 - x_2 = 1, \quad -x_{i-1} + 2x_i - x_{i+1} = 0, \forall i \geq 2.$$

   We verify that $x_i^* = 1 - \frac{i}{n+1}$ satisfies the above equations.

$$\begin{aligned} 2(1 - \frac{1}{n+1}) - (1 - \frac{2}{n+1}) &= 1 \\ -(1 - \frac{i-1}{n+1}) + 2(1 - \frac{i}{n+1}) - (1 - \frac{i+1}{n+1}) &= 0, \forall i \geq 2. \end{aligned}$$

   Thus, the optimal solution x^* is given by

$$x_i^* = 1 - \frac{i}{n+1}, \forall 1 \leq i \leq n.$$

   and the optimal value of the function is given by

$$f(x^*) = \frac{L}{4}(-\frac{1}{2}e_1^T x^*) = \frac{L}{8}(-1 + \frac{1}{(n+1)}).$$

3. We now obtain bounds in order to prove the theorem.

   We have $x_0 = 0$, $\nabla f(x) = Ax - e_1$ and the iterate $x_1 \in \text{span}\{\nabla f(x_0)\} = \text{span}\{e_1\}$. It is easy to see from the structure of A that $\text{span}\{Ae_1, \ldots, Ae_i\} = \text{span}\{e_1, \ldots, e_{i+1}\}$. Thus, we have $x_t \in \text{span}\{e_1, \ldots, e_t\}$.

Using this, we bound $f(x_t)$.

$$f(x_t) \geq \min_{x \in \text{span}\{e_1,\ldots,e_t\}} f(x) = \frac{L}{8}(-1 + \frac{1}{(t+1)}).$$

Now we try to obtain bounds on $\|x_0 - x^*\|^2$. Note that $x_0 = 0$.

$$\begin{aligned} \|x^*\|^2 &= \sum_{i=1}^{n}(1 - \frac{i}{n+1})^2 \\ &= \sum_{i=1}^{n}(\frac{i}{n+1})^2 = \frac{\sum_{i=1}^{n} i^2}{(n+1)^2} \\ &= \frac{n(n+1)(2n+1)}{6(n+1)^2} \\ &\leq \frac{n+1}{3}. \end{aligned}$$

We have used $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

Using the above bounds, we obtain the following,

$$\begin{aligned} \frac{f(x_t) - f(x^*)}{L\|x_0 - x^*\|^2} &\geq \frac{\frac{1}{8}(-1 + \frac{1}{(t+1)}) + \frac{1}{8}(1 - \frac{1}{(n+1)})}{\frac{n+1}{3}} \\ &\geq \frac{\frac{3}{8}(\frac{1}{t+1} - \frac{1}{2t+2})}{2t+2} = \frac{3}{32(t+1)^2}. \end{aligned}$$

We have used $t \leq \frac{1}{2}(n-1)$ which gives $n \geq 2t+1$ in the above to obtain the required bounds.

■

We define l_2 space as the space containing all sequences $x = \{x_i\}_{i=1}^{\infty}$ with finite norm. For convex L smooth functions which are μ strongly convex, we present the following lower bound.

10.2.2 Lower Bound for Smooth and Strongly Convex Problems

Theorem 10.2 *For any $x_0 \in l_2$, and any constants $\mu > 0$ and $L > \mu$, there exists a μ-strongly convex and L-smooth function f such that for any first order method $\mathcal{M}$ that generates a sequence of points x_t such that $x_t \in x_0 + \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_{t-1})\}$, we have*

$$f(x_t) - f(x^*) \geq \frac{\mu}{2}(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^{2t}\|x_0 - x^*\|^2.$$

where $\kappa = \frac{L}{\mu}$.

Proof:

Consider the function f as follows

$$f(x) = \frac{L-\mu}{4}(\frac{1}{2}x^T A x - e_1^T x) + \frac{\mu}{2}\|x\|^2$$

where A and e_1 are as defined in the proof of Theorem 10.1.

1. We show that it is L smooth and μ strongly convex. We have

$$\nabla^2 f(x) = \frac{L-\mu}{4}A + \mu I.$$

In the proof of Theorem 10.1, we have seen that $0 \preceq A \preceq 4I$. Hence we have,

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

2. We now compute the optimal solution of f. From the first order optimality conditions, we have

$$\nabla f(x) = (\frac{L-\mu}{4}A + \mu I)x - (\frac{L-\mu}{4})e_1 = 0$$

which gives $(A + \frac{4}{\kappa-1})x = e_1$. This gives us a set of equations

$$\begin{aligned} 2\frac{\kappa+1}{\kappa-1}x_1 - x_2 &= 1, \\ x_{i+1} + x_{i-1} - 2\frac{\kappa+1}{\kappa-1}x_i &= 0, \forall i \geq 2. \end{aligned}$$

Let $q = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$. We verify that $x_i^* = q^i = (\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^i$ gives us the optimal solution. Substituting q in the optimality conditions gives us

$$q^2 - 2\frac{\kappa+1}{\kappa-1}q + 1 = q(-\frac{(\sqrt{\kappa}+1)^2}{\kappa-1}) + 1 = 0.$$

This solution is also unique due to strong convexity of f.

3. We now find bounds to prove the theorem. From the sum of infinite geometric series, we have,

$$\|x^*\|^2 = \sum_{i=1}^{\infty} q^{2i} = \frac{q^2}{1-q^2}.$$

Similar to the proof of Theorem 10.1, it can be seen that $x_t \in \text{span}\{e_1, \ldots, e_t\}$. Therefore,

$$\|x_t - x^*\|^2 \geq \sum_{i=t+1}^{\infty} q^{2i} = \frac{q^{2(t+1)}}{1-q^2}.$$

From strong convexity of f, we have $f(x_t) - f(x^*) \geq \langle \nabla f(x^*), x_t - x^* \rangle + \frac{\mu}{2}\|x_t - x^*\|^2$. Since x^* is optimal, we have $\langle \nabla f(x^*), x - x^* \rangle \geq 0$ for any x. Hence,

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

We obtain the final result as follows.

$$\frac{f(x_t) - f(x^*)}{\|x_0 - x^*\|^2} \geq \frac{\mu\|x_t - x^*\|^2}{2\|x_0 - x^*\|^2} \geq \frac{\mu}{2}\frac{\frac{q^{2(t+1)}}{1-q^2}}{\frac{q^2}{1-q^2}} = \frac{\mu}{2}q^{2t}.$$

■

Thus, we see that the accelerated gradient descent algorithms discussed in the previous lecture are optimal in the complexity sense.

10.3 Projected Gradient Descent

So far, we were concerned with finding the optimal solution of an unconstrained optimization problem. In real life, optimization problems we are likely to come across constrained optimization problems. In this section, we discuss how to solve constrained optimization problem:

$$\min_{x \in X} f(x)$$

where f is a convex function and X is a convex set.

If we wish to use gradient descent update to a point $x_t \in X$, it is possible that the iterate $x_{t+1} = x_t - \frac{\nabla f(x_t)}{L}$ may not belong to the constraint set X. In the projected gradient descent, we simply choose the point nearest to $x_t - \frac{\nabla f(x_t)}{L}$ in the set X as x_{t+1} i.e., the projection of $x_t - \frac{\nabla f(x_t)}{L}$ onto the set X.

Definition 10.3 *The projection of a point y, onto a set X is defined as*

$$\Pi_X(y) = \operatorname*{argmin}_{x \in X} \frac{1}{2}\|x - y\|_2^2.$$

Projected Gradient Descent (PGD): Given a starting point $x_0 \in X$ and step-size $\gamma > 0$, PGD works as follows until a certain stopping criterion is satisfied,

$$x_{t+1} = x_t - \gamma \Pi_X(x_t - \nabla f(x_t)), \forall t \geq 1.$$

In this lecture, for an L smooth convex function, we fix the step-size to be $\gamma = \frac{1}{L}$.

Proposition 10.4 *The following inequalities hold:*

1. *If $y \in X$, then $\Pi_X(y) = y$.*
2. *The projection onto a convex set X is non-expansive.*
   $\|\Pi_X(x) - \Pi_X(y)\|_2 \leq \|x - y\|_2$,
   $\|\Pi_X(x) - \Pi_X(y)\|_2^2 \leq \langle \Pi_X(x) - \Pi_X(y), x - y \rangle \leq \|x - y\|_2^2$.

Proof:

1. We first note that the $\frac{1}{2}\|x - y\|_2^2$ is strictly convex since $\nabla^2 f(x) = 1$. Hence the solution to the optimization problem is unique. If $y \in X$, then we have $\frac{1}{2}\|y - y\|_2^2 = 0$. Since $\frac{1}{2}\|x - y\|_2^2 \geq 0$, zero is the optimal value of the function and y is it's unique minimizer, thus giving $\Pi_X(y) = y$.
2. For any feasible x^*, the optimality conditions are given by

   $$\langle \nabla f(x^*), z - x^* \rangle \geq 0, \forall z \in X.$$

   Hence for x, y, we have

   $$\langle \Pi_X(y) - y, \Pi_X(x) - \Pi_X(y) \rangle \geq 0,$$
   $$\langle \Pi_X(x) - x, \Pi_X(y) - \Pi_X(x) \rangle \geq 0,$$

   since $\Pi_X(y), \Pi_X(x) \in X$.

Combining the above equations, we have

$$\langle \Pi_X(x) - \Pi_X(y) - (x-y), 2(\Pi_X(y) - \Pi_X(x)) \rangle \geq 0,$$

$$\langle y - x, \Pi_X(y) - \Pi_X(x) \rangle \geq \langle \Pi_X(y) - \Pi_X(x), \Pi_X(y) - \Pi_X(x) \rangle,$$

From Cauchy-Schwartz we further have

$$\langle y - x, \Pi_X(y) - \Pi_X(x) \rangle \leq \|\Pi_X(x) - \Pi_X(y)\|_2 \|x - y\|_2$$

giving us $\|\Pi_X(x) - \Pi_X(y)\|_2 \leq \|x - y\|_2$.

■

10.3.1 Interpretation

We present a few useful observations.

1. Each iterate in the PGD can be viewed as the minimizer of a quadratic approximation of the objective function, similar to the case of Gradient Descent (GD).

   $$x_{t+1} = \underset{x \in X}{\operatorname{argmin}} \{ f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{\mu}{2} \|x - x_t\|^2 \}.$$

   We also observe that the objective function is non-increasing with each iteration, $f(x_{t+1}) \leq f(x_t)$.

2. The optimal solution x^* is a fixed point of $x = \Pi_X(x - \frac{\nabla f(x)}{L})$ i.e., $x^* = \Pi_X(x^* - \frac{\nabla f(x^*)}{L})$.

   Proof:

   $$\begin{aligned}
   x^* \text{ is optimal} &\iff \langle \nabla f(x^*), z - x^* \rangle \geq 0, \forall z \in X \\
   &\iff \langle -\frac{1}{L} \nabla f(x^*), z - x^* \rangle \leq 0, \forall z \in X \\
   &\iff \langle (x^* - \frac{1}{L} \nabla f(x^*)) - x^*, z - x^* \rangle \leq 0, \forall z \in X \\
   &\iff \langle x^* - (x^* - \frac{1}{L} \nabla f(x^*)), z - x^* \rangle \geq 0, \forall z \in X \\
   &\iff x^* \text{ is the projection of } x^* - \frac{1}{L} \nabla f(x^*)
   \end{aligned}$$

   where the last line follows because the projection of y is the minimizer of the function $\frac{1}{2}\|x - y\|_2^2$ over the set X. ■

3. We can rewrite the iterates of PGD as follows

   $$x_{t+1} = x_t - \frac{1}{L} g_X(x_t).$$

   by defining $g_X(x) = L(x - x^\dagger)$, where $x^\dagger = \Pi_X(x - \frac{1}{L} \nabla f(x))$. The function $g_X(x)$ is often called the *Gradient Mapping*. Note that if the problem is unconstrained, $x^\dagger = x - \frac{1}{L} \nabla f(x)$, $g_X(x) = \nabla f(x)$, thus reducing to the usual gradient descent case.

It can be shown that the key inequalities used to show convergence in the gradient descent method follow in a similar way for the projected gradient descent as well by replacing the gradient with the gradient mapping.

10.3.2 Convergence Analysis

Recall that when showing the convergence rates for the gradient descent algorithm, we used the following properties:

(a) For a L-smooth function f, the iterates given by the gradient descent with step size $\gamma = \frac{1}{L}$ satisfy

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

(b) If f is aslo μ-strongly convex, we have

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|^2.$$

Similar results hold for the projected gradient descent which are presented below. Details can be found in Section 2.2 from [NES'04].

Proposition 10.5 *For a convex and L-smooth function f, we have*

$$f(x^\dagger) - f(x) \leq -\frac{\|g_X(x)\|^2}{2L}.$$

If f is also μ-strongly convex, we have

$$\langle g_X(x), x - x^* \rangle \geq \frac{\mu}{2}\|x - x^*\|^2 + \frac{1}{2L}\|g_X(x)\|^2.$$

Remark. With these facts, we can immediately adapt previous convergence analysis for GD method to analyze PGD and obtain similar results, namely, a sublinear rate $O(1/t)$ for general smooth convex case and a linear rate $O((1-\kappa^{-1})^t)$ for the smooth strongly convex case. Moreover, if we combine the projected gradient descent with Nesterov's acceleration, we will also obtain the optimal convergence results for constained convex optimization, similar to what we have in the unconstrained case.

Theorem 10.6 *For a convex function f that is L-smooth, the iterates given by the projected gradient descent with step size $\gamma = \frac{1}{L}$ satisfy*

$$f(x_t) - f(x^*) \leq \frac{2L}{t}\|x_0 - x^*\|^2.$$

If f is further μ-strongly convex, we have

$$\|x_t - x^*\|^2 \leq (1 - \frac{\mu}{L})^t\|x_0 - x^*\|^2.$$

Proof: The proofs are similar to the gradient descent case. We present the proof for μ-strongly convex case.

$$\begin{aligned}
\|x_{t+1} - x^*\|^2 &= \|x_t - \frac{1}{L}g_X(x_t) - x^*\|^2 \\
&= \|x_t - x^*\|^2 + \frac{1}{L^2}\|g_X(x_t)\|^2 - \frac{2}{L}\langle g_X(x_t), x_t - x^* \rangle \\
&\leq \|x_t - x^*\|^2 + \frac{1}{L^2}\|g_X(x_t)\|^2 - \frac{2}{L}(\frac{\mu}{2}\|x_t - x^*\|^2 + \frac{1}{2L}\|\nabla g_X(x_t)\|^2) \\
&= (1 - \frac{\mu}{L})\|x_t - x^*\|^2 \\
&\leq (1 - \frac{\mu}{L})^t\|x_0 - x^*\|^2.
\end{aligned}$$

where we have used Proposition 10.5 to bound the inner product. ■

10.3.3 Examples

We present a few examples to illustrate how to find projection onto different sets.

1. l_2 **ball**: $X = \{x : \|x\|_2 \leq 1\}$. The projection is given by

$$\Pi_X(y) = \begin{cases} y & \text{for } \|x\|_2 \leq 1, \\ \frac{y}{\|y\|_2} & \text{for } \|x\|_2 > 1. \end{cases}$$

2. **Box**: $X = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$. The i^{th} coordinate of the projection, $\forall 1 \leq i \leq n$, is given by

$$[\Pi_X(y)]_i = \begin{cases} \ell_i & \text{for } x_i < \ell_i, \\ x_i & \text{for } \ell_i \leq x_i \leq u_i, \\ u_i & \text{for } x_i > u_i. \end{cases}$$

3. **Simplex**: $X = \{x \in \mathbb{R}^n : x_i \geq 0, \sum_{i=1}^n x_i \leq 1\}$. Computing the projection requires solving the quadratic problem

$$\Pi_X(y) = \underset{x_i \geq 0; \sum x_i \leq 1}{\operatorname{argmin}} \frac{1}{2}\|x - y\|_2^2.$$

   If $y \in X$, then $\Pi_X(y) = y$. Let's consider the case when $y \notin X$. We form the Langrangian dual function for $\lambda \geq 0$ and $\mu \geq 0$,

$$L(x, \lambda, \mu) = \frac{1}{2}\|x - y\|_2^2 + \lambda^T(-x) + \mu(\sum x_i - 1).$$

   We use KKT conditions in order to solve the optimization problem. The optimal solution satisfies

$$\begin{aligned} x_i - y_i - \lambda_i + \mu = 0, \forall i & \quad \text{(optimality)} \\ \lambda^T x = 0 & \quad \text{(complementary slackness)} \\ \mu(\sum x_i - 1) = 0 & \quad \text{(complementary slackness)} \\ \lambda \geq 0, \mu \geq 0 & \quad \text{(feasibility)} \\ x \geq 0, \sum_i x_i \leq 1 & \quad \text{(feasibility)} \end{aligned}$$

   Let μ^* be such that $\sum(y_i - \mu^*)_+ = 1$, where $(x)_+ = \max\{x, 0\}$. Let $x_i^* = (y_i - \mu^*)_+$ and $\lambda_i = x_i^* - (y_i - \mu^*)$ for any i. We can easily verify that the solution $(x^*; \lambda^*, \mu^*)$ satisfies the KKT conditions listed above. Therefore, the projection of y onto the simplex set X is given by,

$$\Pi_X(y) = \begin{cases} y & \text{for } y \geq 0, \sum y_i \geq 0, \\ (y - \mu^*)_+ & \text{otherwise.} \end{cases}$$

   where μ^* is the solution to the constraint $\sum(y_i - \mu)_+ = 1$. Note that μ^* can be found using tools such as the bisection method.

4. **Nuclear ball**: $X = \{x \in \mathbb{R}^{m \times n} : \|x\|_{\text{nuc}} = \sum \sigma_i(x) = \|\sigma(x)\|_1 \leq 1\}$.

   From the singular value decomposition of y, there exist appropriate matrices U, V, Σ such that $y = U\Sigma V^T = \sum \sigma_i u_i v_i^T$.

Let $x = Ux'V^T$. Since we are looking for the projection of a diagonal matrix Σ, the closest x' will be a diagonal matrix and well thus x' denotes the singular values of x. Let the singular values of y be denoted by a vector $\sigma \in \mathbb{R}^{\text{rank}(y)}$ and that of x' by $\sigma' \in \mathbb{R}^{\text{rank}(y)}$.

$$\Pi_X(y) = \underset{\|x\|_{nuc} \leq 1}{\text{argmin}} \frac{1}{2}\|x - y\|_2^2 = \underset{\sigma' \geq 0; \sum \sigma'_i \leq 1}{\text{argmin}} \frac{1}{2}\|\sigma' - \sigma\|_2^2.$$

This is equivalent to projecting onto a simplex. Therefore, the projection of y onto the nuclear ball is given by,

$$\Pi_X(y) = \begin{cases} y & \text{for } \|\sigma(y)\|_1 \leq 1, \\ \sum(\sigma_i - \mu^*)_+ u_i v_i^T & \text{otherwise.} \end{cases}$$

where μ^* is the solution to the constraint $\sum(\sigma_i - \mu)_+ = 1$. Note that in order to compute the projection, we need to find the full singular value decomposition of an $m \times n$ matrix, the complexity of which is known as $O(mn^2)$. Hence using projected gradient descent may be computationally prohibitive for high-dimensional problems.

References

[NES '04] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer, 2004.

IE 598: Big Data Optimization **Fall 2016**

Lecture 11: Conditional Gradient (Frank-Wolfe Algorithm) - September 27

Lecturer: Niao He *Scriber: Cheng-Yang (Peter) Liu*

Overview: In this lecture we are going to discuss a new gradient-based and projection-free algorithm, called Conditional Gradient Method, also known as Frank-Wolfe Algorithm. The algorithm was initially proposed by Frank and Wolfe in 1956 for solving quadratic programming problems with linear constraints, but regained many interests and new developments in machine learning community over the last few days. In this lecture, we are going to discuss the motivation, algorithm itself, convergence analysis, lower bound, and some recent advances.

11.1 Introduction

In the previous lecture, we discussed how to solve constrained convex optimization problems

$$\min_{x\in X} f(x)$$

by Projected Gradient Descent, which requires to compute the projection on a convex set at each iteration. Recall the definition of projection of a vector y on a convex set X is defined as $\Pi_X(y) = \text{argmin}_{x\in X} \frac{1}{2}\|y-x\|_2^2$. Here are some examples.

- **Simplex:** $X = \{x \in \mathbb{R}^n : x \geq 0, \sum x_i \leq 1\}$. The projection is given by

$$\Pi_X(y) = \begin{cases} y & y \in X \\ (y-\mu_\star)_+ & y \notin X \end{cases}$$

  where μ_* is the Lagrange multiplier such that $\sum_i (y_i - \mu_*)_+ = 1$.
  The simplex domain is widely used in many applications including portfolio management, LASSO, boosting, MAP inference, density estimation, structured SVM, etc.

- **Nuclear norm ball:** $X = \{x \in \mathbb{R}^{m\times n}, \|x\|_{nuc} = \|\sigma(x)\|_1 \leq 1\}$. The projection is given by

$$\Pi_X(y) = \begin{cases} y & y \in X \\ \sum_i (\sigma_i - \mu_\star)_+ u_i v_i^T & y \notin X, \quad \text{where } y = \sum_i \sigma_i u_i v_i^T \end{cases}$$

  where μ_* is the Lagrange multiplier such that $\sum_i (\sigma_i - \mu_*)_+ = 1$.
  The nuclear norm domain is widely used in many recent applications including robust PCA, matrix completion, network estimation, etc.

- **Polyhedron:** $X = \{x \in \mathbb{R}^n : Ax \leq b\}$. The projection in this case simply reduces to solving a convex quadratic program (QP).

Apparently, there are advantages and disadvantages when using Projected Gradient Descent. The pros are:

(i) In many cases, computing the projection often admits closed-form solution or easy-to-solve subproblems, e.g. ℓ_2-ball, convex cone, halfspaces, box, simplex, etc.

(ii) Projected Gradient Descent enjoyes the same convergence rates and similar analysis as the unconstrained case.

On the other hand, the potential cons are that:

(i) For high-dimensional problems, computing the projection can be computational expensive. e.g., it requires $\mathcal{O}(n)$ for simplex domain, $\mathcal{O}(mn^2 \wedge nm^2)$ for nuclear norm domain, and iteratively solving a QP for polyhedron domains.

(ii) Full gradient updating may destroy certain structure, such as sparsity and low rank.

To avoid the expense of projection, an alternative is to use linear minimization over the convex set instead!

Definition 11.1 (Linear minimization oracle) *For a given vector y, the linear minimization oracle over a convex set X is defined as*

$$LMO_X(y) = \operatorname*{argmin}_{x \in X} y^T x.$$

Let us revisit those examples.

- **Simplex:** $X = \{x \in \mathbb{R}^n : x \geq 0, \sum x_i = 1\}$. The linear minimization oracle is given by

  $$\text{LMO}_X(y) = e_i, \quad i = \operatorname*{argmin}_{k=1,\ldots,n}(y_k)$$

  which has only one non-zero entry and is much simpler than projection.

- **Nuclear norm ball:** $X = \{x \in \mathbb{R}^{m \times n}, \|x\|_{nuc} = \|\sigma(x)\|_1 \leq 1\}$. The linear minimization oracle is given by

  $$\text{LMO}_X(y) = -u_i v_i^T, \quad i = \operatorname*{argmax}_{k=1,\ldots,n}(\sigma_k)$$

  which is a rank one matrix and requires only to calculate the top singular value and corresponding singular vectors. This can be efficiently done via power iteration.

- **Polyhedron:** $X = \{x \in \mathbb{R}^n : Ax \leq b\}$. The linear minimization reduces to solving a linear program (LP) instead of QP.

11.2 Generic Conditional Gradient Method

We consider the constrained convex optimization problem

$$\min_{x \in X} f(x)$$

where f is L-smooth and convex, set X is convex and compact. We denote the diameter of the set X as $D = \max_{x,y \in X} \|x - y\|_2$, which is finite since X is compact.

Conditional Gradient method. The algorithm works as follows. We start with an initial solution x_0 and at each iteration, compute the solution $\hat{x}_t$ which minimizes the linear approximation $f(x_t)+\nabla f(x_t)^T(x-x_t)$ of objective function. Then, we update x_t to x_{t+1} such that it is at least as good as the intermediate point $\gamma_t\hat{x}_t + (1-\gamma_t)x_t$, namely, $f(x_{t+1}) \leq f(\gamma_t\hat{x}_t + (1-\gamma_t)x_t)$.

- *Algorithm*:
  - Initialize $x_0 \in X$
  - Compute $\hat{x}_t = \text{LMO}_X(\nabla f(x_t)) = \text{argmin}_{x\in X}\langle x, \nabla f(x_t)\rangle$
  - Find $x_{t+1} \in X$, s.t. $f(x_{t+1}) \leq f(\gamma_t\hat{x}_t + (1-\gamma_t)x_t)$, with $\gamma_t = \frac{2}{t+2}$

  There are several different ways to update x_{t+1} at each iteration.
- *Updating rules*
  - fixed step size: $x_{t+1} = \gamma_t\hat{x}_t + (1-\gamma_t)x_t$
  - line search: $x_{t+1} = \text{argmin}_{0\leq\gamma\leq 1} f(\gamma\hat{x}_t + (1-\gamma)x_t)$
  - fully-corrective search: $x_{t+1} = \text{argmin}_{x\in conv\{x_0,\hat{x}_1,\ldots,\hat{x}_t)} f(x)$

  There is clearly some tradeoff between the computation cost of updating x_{t+1} vs. the reduction of objective function value. The more effort you pay for each iteration, most likely you will get more reduction, and the algorithm will converge faster.

In contrast to Projected Gradient Descent, Conditional Gradient method requires cheaper iteration cost and maintains desirable structure of the solution such as sparsity and low rank.

11.3 Convergence and Lower Bound Complexity

Theorem 11.2 (Convergence) *Let f be L-smooth and convex, X be convex compact with diameter $D > 0$. The above Conditional Gradient method satisfies*

$$f(x_{t+1}) - f(x_*) \leq \frac{2LD^2}{t+2}, \quad \forall t \geq 1$$

Proof: By convexity, we have

$$f(x) \geq f(x_t) + \nabla f(x_t)^T(x - x_t), \forall x \in X.$$

Taking minimum on both side leads to

$$f(x_*) \geq f(x_t) + \nabla f(x_t)^T(\hat{x}_t - x_t).$$

Hence, we have

$$f(x_t) - f(x_*) \leq \nabla f(x_t)^T(x_t - \hat{x}_t).$$

Let us denote $\tilde{x}_{t+1} = \gamma_t \hat{x}_t + (1-\gamma_t)x_t$ as the intermediate point. By definition of x_{t+1}, we have

$$\begin{aligned} f(x_{t+1}) - f(x_t) &\leq f(\tilde{x}_{t+1}) - f(x_*) \\ &\leq f(x_t) + \nabla f(x_t)^T(\tilde{x}_{t+1} - x_t) + \tfrac{L}{2}\|\tilde{x}_{t+1} - x_t\|^2 - f(x_*) \\ &\leq f(x_t) + \gamma_t \nabla f(x_t)^T(\hat{x}_t - x_t) + \tfrac{L\gamma_t^2}{2}\|\hat{x}_t - x_t\|_2^2 - f(x_*) \\ &\leq f(x_t) - f(x_*) - \gamma_t(f(x_t) - f(x_*)) + \tfrac{L{\gamma_t}^2}{2}D^2 \end{aligned}$$

Define $\epsilon_t = f(x_t) - f_*$, we arrive at

$$\epsilon_{t+1} \leq (1-\gamma_t)\epsilon_t + \frac{L{\gamma_t}^2}{2}D^2$$

By induction, we can show that $\epsilon_t \leq \frac{2LD^2}{t+2}$. First of all, when $t = 1$, $\gamma_0 = 1$, we have

$$\epsilon_1 \leq (1-\gamma_0)\epsilon_0 + \frac{LD^2}{2}\gamma_0^2 = \frac{LD^2}{2} \leq \frac{3}{2}LD^2.$$

Now assume it holds for $t \geq 1$, that $\epsilon_t \leq \frac{2LD^2}{t+2}$, then

$$\begin{aligned} \epsilon_{t+1} &\leq (1 - \tfrac{2}{t+2}) \cdot \tfrac{2LD^2}{t+2} + \tfrac{LD^2}{2}(\tfrac{2}{t+2})^2 \\ &= \tfrac{2LD^2(t+1)}{(t+2)^2} \\ &\leq \tfrac{2LD^2}{t+3}, \quad \text{since } (t+2)^2 \geq (t+1)(t+3). \end{aligned}$$

■

Remark. The above result implies that for smooth convex problems, Conditional Gradient method achieves an overall $O(LD^2/t)$ convergence rate. Recall that when applied with Nesterov's accelerated gradient descent, we obtain a $O(LD^2/t^2)$ rate for smooth convex problems and a linear $O((\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}})^t)$ rate for smooth strongly convex problems. Does this imply conditional gradient is sub-optimal? Can we improve the convergence results for strongly convex problems? The following theorem on lower bound complexity suggests that the answer is no.

Theorem 11.3 (Lower Bound) *For any $x_0 \in \mathbb{R}^n$, $\quad 1 \leq t \leq \frac{n}{2} - 1$, then $\exists$ a L-smooth convex function f and a convex set X with diameter D s.t. for any algorithm $\mathcal{M}$ that only computes local gradients of f and do linear minimization over X, we have*

$$f(x_t) - f_* \geq \frac{LD^2}{8(t+1)}, \ \forall t \geq 1.$$

Proof: We construct the case when

$$f(x) = \frac{L}{2}\|x\|_2^2$$

$$X = \{x \in \mathbb{R}^n : x \geq 0, \sum x_i = D_0\}, \text{ where } D_0 = \frac{D}{\sqrt{2}}.$$

Hence, f is L-smooth and convex, X is convex compact with diameter D. Consider the optimization problem

$$\min_{x \in X} f(x),$$

the optimal solution is given by $x_* = (\frac{D_0}{n}, \ldots, \frac{D_0}{n})$ and the optimal value is $f(x_*) = \frac{LD_0^2}{2n}$.

Without loss of generality, let $x_0 = D_0 e_1$. Hence, by induction, we can see that

$$x_t = conv(D_0 e_1, D_0 e_{p_1}, \ldots, D_0 e_{p_t})$$

for some index $p_1, \ldots, p_t$. Therefore,

$$f(x_t) \geq \min_{x \in D_0 conv(e_0, e_{p_1}, \ldots, e_{p_t})} f(x) \geq \frac{LD_0^2}{2(t+1)}$$

Since $n \geq 2(t+1)$, we further have

$$\begin{aligned} f(x_t) - f(x_*) &\geq \frac{LD_0^2}{2}(\frac{1}{t+1} - \frac{1}{n}) \\ &\geq \frac{LD_0^2}{2}(\frac{1}{t+1} - \frac{1}{2(t+1)}) \\ &\geq \frac{LD_0^2}{4(t+1)} = \frac{LD^2}{8(t+1)} \end{aligned}$$

which proves the theorem. ■

Remark. The above theorem implies that

1. CG is indeed optimal among all gradient-based algorithms that only use linear minimization oracles over the domain.
2. The convergence rate $\mathcal{O}(\frac{1}{t})$ cannot be improved even assuming strong convexity.

Recent advances. We mention a few recent works on conditional gradient methods here:

1. **Linear convergence for some strongly convex problems**
   Although we show that in the worst case, we cannot improve the sublinear rate for strongly convex cases, it is shown that for specific situation (e.g., domain is given by a polyhedron) or under additional assumptions (e.g., optimum lies in the interior), linear rate of convergence can be established. See recent works [BS16] and [LJ15].
2. **Improvement over gradient oracles**
   Note that in order to achieve an ϵ-accuracy solution, the conditional gradient requires $\mathcal{O}(1/\epsilon)$ number of calls to compute the gradient. A recent algorithm, called *conditional gradient sliding*, is able to improve this to optimal complexity bounds, i.e., $\mathcal{O}(1/\sqrt{\epsilon})$ for smooth convex problems and $\mathcal{O}(\log(\frac{1}{\epsilon}))$ for smooth strongly convex problems, respectively. See details in [LY14].

References

[J13] M. Jaggi, "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization," *ICML*, 2013.

[HJN14] Z. Harcaoui, A. Juditsky and A. Nemirovski, "Conditional gradient algorithms for norm-regularized smooth convex optimization," *Mathematical Programming*, 1–32, 2014.

[BS16] A. Beck and S. Shtern, "Linearly convergent away-step conditional gradient for non-strongly convex functions," *Mathematical Programming*, 1–27, 2016.

[LJ15] S. Lacoste-Julien and M. Jaggi, "On the Global Linear Convergence of Frank-Wolfe Optimization Variants," *NIPS*, 2015.

[GH12] D. Garber and E. Hazan, "Playing non-linear games with linear oracles," *In 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, 2013a.

[LY14] G. Lan and Z. Yi, "Conditional gradient sliding for convex optimization," *Optimization-Online preprint*, 4605, 2014.

IE 598: Big Data Optimization **Fall 2016**

Lecture 12: Coordinate Descent Algorithms

Lecturer: Niao He *Scriber: Lucas Buccafusca*

Overview: In this lecture we are going to discuss a new family of algorithm, called Coordinate Descent Methods. We will dicuss the general frameowrk and three diffirent variants of block coordinate gradient descent (Gauss Southwell, Randomized, and Cyclic) by emloying diffirent updating rules.

12.1 Introduction: Coordinate Descent Algorithms

Once again, the goal is to solve

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

where f is convex and smooth (f is continuously differentiable and gradient is Lipschitz continuous).

Motivation. As discussed in previous lectures, when n is large, it becomes computationally expensive to calculate full gradients, which means gradient descent is not necessarily always efficient. Observe that for unconstrained problems, x_* is an optimal solution if and only if $\nabla f(x_*) = 0$, namely, $\nabla_i f(x_*) = 0, \forall i = 1, \ldots, n$. To find the optimal solution, it makes sense to search along each coordinate direction; if at some point, the objective is not decreasing at every coordinate direction, then we have reached the optimum. This motivates the so called *Coordinate Minimization algorithms*, or also called *Coordinate Descent algorithms*.

Coordinate descent algorithms are derivative-free optimization methods.

Coordinate Minimization The general idea for a coordinate descent algorithm is shown below

Initialize $x^{(0)}$
  for $t = 1, 2, \ldots$
    Pick coordinate i from $1, 2, \ldots n$
    $x_i^{(t+1)} = \text{argmin}_{x_i\in\mathbb{R}} f(x_i, \omega_{-i}^t)$
  end

where ω_{-i}^t represent all other coordinates and are specified below.

The above framework is rather general, and there are many ways to choose ω_{-i}^t and to choose the coordinates to update at each iteration.

Rules for updating. We mention two styles here:

- **Gauss – Seidel style**

  $$\omega_{-i}^t = (x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)})$$

  When updating each coordinate, the Gauss-Seidel style fixes the rest coordinates to be the most up-to-date solution, with a potential advantage of converging faster.

- **Jacobi style**

$$\omega_{-i}^{t} = (x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)})$$

When updating each coordinate, the Jacobi style fixes the rest coordinates to be the solution obtained from previous cycle, i.e. it doesn't care about intermediary iterates until we complete all coordinates. The Jacobi update has a unique advantage; we can run the update for each coordinate in parallel.

Rules for selecting coordinates There are several ways and orders to decide which coordinates to update at each iteration.

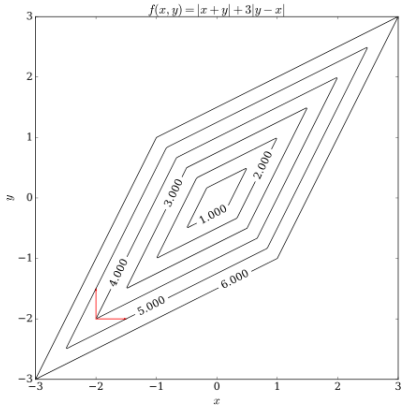
- **Cyclic Order** : run all coordinates in cyclic order, i.e. $1 \to 2 \to 3 \to \ldots \to n$ at each iteration
- **Random Sampling** : Randomly select some coordinate to update (sample with replacement)
- **Gauss−Southwell** : At each iteration, pick coordinate i so that $i = \underset{1 \le j \le n}{\operatorname{argmax}} |\nabla_j f(x^{(t)})|$
- **Random Permutation** : run cyclic order on a permuted index (sample without replacement)

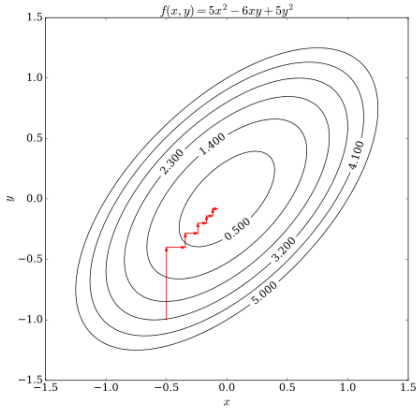
For example if $n = 3$ we could have the following:
Cyclic: $(1 \to 2 \to 3) \to (1 \to 2 \to 3) \to (1 \to 2 \to 3) \to \ldots$
Random permutation : $(2 \to 1 \to 3) \to (3 \to 1 \to 2) \to (1 \to 2 \to 3) \to \ldots$
Random sampling: $1 \to 1 \to 2 \to 3 \to 2 \to 1 \to \ldots$

Note that except for the Gauss-Southwell case, the general Coordinate Descemt algorithm can be seen as a derivative-free algorithm. Here are some immediate observations.

Remarks on Coordinate Descent

1. The objective function values are non-decreasing: $f(x^{(0)}) \ge f(x^{(1)}) \ge \ldots$
2. If f is strictly convex and smooth, the algorithm converges to a global minimum (optimal solution)
3. If f is non-convex or not even smooth, the algorithm might not converge at all.
   Example: $f(x, y) = |x + y| + 3|x - y|$.
   If started with $(x^0, y^0) = (-1, -1)$, the algorithm will not move. [1]





[1] Example and image from https://en.wikipedia.org/wiki/Coordinate descent

Suppose that the algorithm is on a corner point in the non-smooth case; then there are two directions it can try, indicated by the red arrows. However, every step along these two directions will increase the objective function's value, so the algorithm will not take a step, even though the sum of both steps would bring the algorithm closer to the optimum. We see strict convergence for the smooth case on the right.

The framework can be generalized for block updates e.g. you split your decision variable into blocks and you can cyclically update each block. This is usually known as *block coordinate descent*. In the case when we have tow blocks, block coordinate descent simply reduces to the *alternating minimization*.

Despite of its popularity, the global convergence rates of coordinate descent algorithms have only been recently established and most analysis focus on block coordinate gradient descent type of algorithms.

- CD with random sampling [Nesterov, 2010] and [Richtarik and Takac, 2011]
- CD with cyclic order and random permutation [Beck and Tetruashvili, 2013] and [Hong et.al., 2014]
- CD with Gauss-Southwell update [Nesterov, 2010] and [Nutini et.al., 2015]

12.2 Several Coordinate Descent Algorithms and Convergences

Consider the unconstrained optimization problem with block structure:

$$\min_{x=[x_1;\ldots;x_b]\in\mathbb{R}^{n_1}\times\ldots\mathbb{R}^{n_b}} f(x)$$

where $n_1+\ldots+n_b=n$. Let us decompose the identity matrix $I_{n\times n}=[U_1|\ldots|U_b]$, where U_i is a matrix of size n by n_i with diagonals being 1 and 0 elsewhere. Hence, for any vector x, we have $x=\sum_{i=1}^{b}U_ix_i$, and $x_i=U_i^Tx$.

We assume that

1. f is L-smooth, $||\nabla f(x)-\nabla f(y)||_2\leq L||x-y||_2$ for some Lipschitz constant $L\geq 0$.

2. $f(\cdot,x_{-i})$ is L_i-smooth, i.e. $||\nabla f(x+U_ih_i)-\nabla f(x)||_2\leq L_i||h_i||_2,\ \forall h_i\in\mathbb{R}^{n_i}$ and $x\in\mathbb{R}^n$.

Denote $L_{max}=\max_i L_i$ and $L_{min}=\min_i L_i$. We have $L_i\leq L\leq\sum_i L_i\leq b\cdot L_{max},\forall i=1,\ldots,b$.

12.2.1 Gauss-Southwell Block Coordinate Gradient Descent

GS-BCGD At iteration t, pick index $i_t=\underset{1\leq j\leq n}{\operatorname{argmax}}|\nabla_j f(x^{(t)})|$

$$x^{(t+1)}=x^{(t)}-\frac{1}{L}U_{i_t}\nabla_{i_t}f(x^{(t)})$$

Here, we use a fixed step-size of $\frac{1}{L}$ at each iteration.

Theorem 12.1 *If f is convex and L-smooth, we have*

$$f(x^{(t)}) - f^* \leq \frac{2Lb||x^0 - x^*||^2}{t}$$

If f is μ-strongly convex and L-smooth, then

$$f(x^{(t)}) - f^* \leq (1 - \frac{\mu}{bL})^t (f(x^0) - f^*)$$

Proof: We have

$$f(x^{(t)}) - f(x^{(t+1)}) \geq \frac{1}{2L}||\nabla f_{i_t}(x^{(t)})||_2^2 \geq \frac{1}{2Lb}||\nabla f(x^{(t)})||_2^2.$$

From the previous analysis of gradient descent, we see that the only difference is an additional b term that propagates through. Hence, following same arguments leads to the convergence results. ■

12.2.2 Randomized Block Coordinate Gradient Descent

Randomized-BCGD At iteration t, pick index i_t uniformly randomly from $\{1, \ldots, b\}$
$$x^{(t+1)} = x^{(t)} - \frac{1}{L} U_{i_t} \nabla_{i_t} f(x^{(t)})$$

Note that the index i_t is a discrete random variable, and $P(i_t = i) = \frac{1}{b}, i = 1, \ldots, b$

Theorem 12.2 *If f is convex and L-smooth, we have*

$$\mathbb{E}[f(x^{(t)}) - f^*] \leq \frac{2Lb \cdot R(x_0)^2}{t}$$

where $R(x_0) = \max\{||x - x^*|| : f(x) \leq f(x_0)\}$. If f is μ-strongly convex and L-smooth, then

$$\mathbb{E}[f(x^{(t)}) - f^*] \leq (1 - \frac{\mu}{bL})^t (f(x^0) - f^*)$$

Proof: In this case, we have

$$f(x^{(t)}) - f(x^{(t+1)}) \geq \frac{1}{2L}||\nabla f_{i_t}(x^{(t)})||_2^2$$

Taking expectation on both side over i_t: we have

$$f(x^{(t)}) - \mathbb{E}_{i_t} f(x^{(t+1)}) \geq \frac{1}{2L}\mathbb{E}_{i_t}||\nabla f_{i_t}(x^{(t)})||_2^2$$

Since $P(i_t = i) = \frac{1}{b}$, thus

$$f(x^{(t)}) - \mathbb{E}_{i_t} f(x^{(t+1)}) \geq \frac{1}{2L} \sum ||\nabla f_i(x^{(t)})||_2^2 P(i_t = i) = \frac{1}{2Lb}||\nabla f(x^{(t)})||_2^2$$

Similarly to above, we see that the differences are the expectation and the factor b. Another difference is that we will no longer have

$$f(x^{(t)}) - f^* \leq \|x^0 - x^*\| \cdot \|\nabla f(x^{(t)})\|_2$$

since we cnanot ensure that $\|x^0 - x^*\| \geq \|x^t - x^*\| \geq \|x^{t+1} - x^*\|$. But instead we have

$$f(x^{(t)}) - f^* \leq R(x_0)\|\nabla f(x^{(t)})\|_2.$$

Combining these two facts lead to the convergence results. ■

Remarks. There are several ways to improve the rate:

- Use larger step-sizes (use $\gamma_t = \frac{1}{L_{i_t}}$ instead of $\gamma_t = \frac{1}{L}$).
- Use non-uniform sampling distribution (use $P_i = \frac{L_i}{\sum L_i}$ instead of $P_i = \frac{1}{b}$) i.e. choose blocks with larger Lipschitz constants more frequently.

This gives rise to an improved convergence bound (see details in [Nesterov, 2010] and [Richtarik and Takac, 2011])

$$\mathbb{E}[f(x^{(t)}) - f^*] \leq \frac{2\sum_{i=1}^{b} L_i \cdot R(x_0)^2}{t}$$

For instance, in the case when $L_i = \frac{L}{b}, i = 1, \ldots, b$, the randomized block coordinate descent algorithm (Randomized-BCGD) achieves the same complexity result as the full gradient descent (GD) algorithm, but the iteration cost is $O(b)$ times cheaper.

12.2.3 Cyclic Block Coordinate Gradient Descent

Cyclic-BCGD	At iteration t, for $i = 1 \ldots b$, do
	$x_i^{(t)} = x_{i-1}^{(t)} - \frac{1}{L} U_i \nabla_i f(x_{i-1}^{(t)})$
	Set $x^{(t+1)} = x_b^{(t+1)}$

Note that the Cyclic-BCGD is a deterministic algorithm, and the iteration cost for Cyclic-BCGD is $O(b)$ times larger than the previous Randomized-BCGD since it needs to go through entire blocks.

Theorem 12.3 *If f is convex and L-smooth, we have*

$$f(x^{(t)}) - f^* \leq \frac{4L(b+1)R(x_0)^2}{t}.$$

If f is μ-strongly convex and L-smooth, it converges linearly:

$$f(x^{(t)}) - f^* \leq (1 - \frac{\mu}{2(b+1)L})^t (f(x^0) - f^*)$$

Proof: In this case, we have

$$f(x^{(t)}) - f(x^{(t+1)}) \geq \sum_{i=1}^{b} (f(x_{i-1}^{(t)}) - f(x_i^{(t)})) \geq \sum_{i=1}^{b} \frac{1}{2L} ||\nabla_i f(x_{i-1}^{(t)})||_2^2)$$

We now claim that:

$$||\nabla f(x^{(t)})||_2^2 \leq (2b+2) \sum_{i=1}^{b} ||\nabla_i f(x_{i-1}^{(t)})||_2^2$$

To prove the claim, we first observe that

$$||\nabla_i f(x^{(t)})||_2^2 \leq 2||\nabla_i f(x_{i-1}^{(t)}) - \nabla_i f(x^{(t)})||_2^2 + 2||\nabla_i f(x_{i-1}^{(t)})||_2^2$$

since $||a+b||^2 \leq 2||a||^2 + 2||b||^2$.

We now use the Lipschitz conditions to simplify to:

$$||\nabla_i f(x^{(t)})||_2^2 \leq 2L^2||x_{i-1}^{(t)} - x^{(t)}||^2 + 2||\nabla_i f(x_{i-1}^{(t)})||^2 = 2L^2||\sum_{j=1}^{i-1} \tfrac{1}{L} U_j \nabla_j f(x_{j-1}^{(t)})||^2 + 2||\nabla_i f(x_{i-1}^{(t)})||^2$$

Cancelling terms:

$$||\nabla_i f(x^{(t)})||_2^2 \leq 2||\sum_{j=1}^{b} \nabla_j f(x_{j-1}^{(t)})||^2 + 2||\nabla_i f(x_{i-1}^{(t)})||^2$$

Taking summation over $i = 1, \ldots, b$ and invoking the fact that

$$||\nabla f(x^{(t)})||_2^2 = \sum_{i=1}^{b} ||\nabla_i f(x^{(t)})||_2^2$$

we prove the claim.

This now gives us:

$$f(x^{(t)}) - f(x^{(t+1)}) \geq \tfrac{1}{4L(b+1)} ||\nabla f(x^{(t)})||_2^2$$

From the previous analysis of gradient descent, we arrive at the stated results. ■

Remark. If a larger stepsize $\gamma_i = \frac{1}{L_i}$ is adopted, the rate can be improved to

$$f(x^{(t)}) - f^* \leq \frac{4L_{max}(bL^2/L_{min}^2 + 1)R(x_0)^2}{t}.$$

where $L_{max} = \max_i L_i$ and $L_{min} = \min_i L_i$.

Summary. Here is brief summary of the converge rates for the three variants of block coordinate gradient descent algorithms:

Method	$\gamma_t = \frac{1}{L}$	$\gamma_t = \frac{1}{L_i}$
Gauss-Southwell	$\mathcal{O}(\frac{Lb}{t})$	$\mathcal{O}(\frac{L_{max}b}{t})$
Cyclic	$\mathcal{O}(\frac{Lb}{t})$	$\mathcal{O}(\frac{L_{max}(bL^2/L_{min}^2+1)}{t})$
Randomized	$\mathcal{O}(\frac{Lb}{t})$	$\mathcal{O}(\frac{\sum L_i}{t})$

Some open questions still remain about these complexity bounds. For example, are these bounds optimal, or under which situation can we drop the dependence on b, under which regime is cyclic RBGD better than randomized RBGD? In practice, their behaviors could vary a lot depending on the choice of step-sizes and the underlying parameters of problem instances.

References

[1] Nesterov, Yu , "Efficiency of coordinate descent methods on huge-scale optimization problems." SIAM Journal on Optimization 22.2 (2012): 341-362.

[2] Richtarik, Peter, and Martin Takac. "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function." Mathematical Programming 144.1-2 (2014): 1-38.

[3] Beck, Amir, and Luba Tetruashvili. "On the convergence of block coordinate descent type methods." SIAM Journal on Optimization 23.4 (2013): 2037-2060.

[4] Hong, Mingyi, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. "Iteration complexity analysis of block coordinate descent methods." arXiv preprint arXiv:1310.6957 (2013).

[5] Nutini, Julie, Mark Schmidt, Issam H. Laradji, Michael Friedlander, and Hoyt Koepke. "Coordinate descent converges faster with the Gauss-Southwell rule than random selection." Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 2015.

IE 598: Big Data Optimization **Fall 2016**

Lecture 13: Case Study on Logistic Regression – October 06

Lecturer: Niao He *Scriber: Jialin Song*

Overview: In this lecture we summarize algorithms we have covered so far for smooth convex optimization problems. To better understanding the pros and cons of different algorithms, we conduct a case study on logistic regression model to investigate the their empirical performances.

13.1 Summary

We summarize performance of algorithms on smooth convex optimization (including Interior Point Method (IPM), Gradient Descent (GD), Accelerated Gradient Descent (AGD), Projection Gradient Descent (PGD), Frank-Wolfe Algorithm (FW), Block Coordinated Gradient Descent (BCGD)) in terms of convergence rate, iteration complexity and iteration cost in Table 13.1.

Table 13.1: Performance of algorithms on structured convex optimization and smooth convex optimization

	Algorithm	Convergence Rate		Iteration Complexity		Iteration Cost
Structured Convex Optimization (LP/SOCP/DCP)	IPM	$O(\nu \exp(-\frac{t}{\sqrt{\nu}}))$		$O(\sqrt{\nu}\log(\frac{1}{\epsilon}))$		one Newton step
		Convex	Strongly Convex	Convex	Strongly Convex	
Smooth Convex Optimization	GD	$O(\frac{LD^2}{t})$	$O((\frac{1-\kappa}{1+\kappa})^{2t})$	$O(\frac{LD^2}{\epsilon})$	$\frac{L}{\mu}\log(\frac{1}{\epsilon})$	one gradient
	AGD	$O(\frac{LD^2}{t^2})$	$O((\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}})^{2t})$	$O(\frac{\sqrt{L}D}{\sqrt{\epsilon}})$	$O(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$	one gradient
	PGD	$O(\frac{LD^2}{t})$	$O((1-\frac{\mu}{L})^t)$	$O(\frac{LD^2}{\epsilon})$	$\frac{L}{\mu}\log(\frac{1}{\epsilon})$	one gradient one projection
	FW	$O(\frac{LD^2}{t})$		$O(\frac{LD^2}{\epsilon})$		one gradient one linear minimization
	BCGD	$O(\frac{bLD^2)}{t}$	$O((1-\frac{\mu}{bL})^t)$	$O(\frac{bLD^2}{\epsilon})$	$O(\frac{bL}{\mu}log(\frac{1}{\epsilon}))$	$O(1)$-block gradient (randomized) $O(b)$-block gradient (cyclic) $O(b)$-block gradient (Gauss Southwell)

Notations:

- L is L-smooth constant
- μ is strong convexity parameter
- κ=$\frac{L}{\mu}$ ibs the condition number
- D is either $\|x_0 - x_*\|_2$ or diameter of compact set X.

13.2 Logistic Regression

In this section, we first introduce the basic fundamentals of classification models and then formulate logistic regression model.

13.2.1 Preliminaries on Classification Model

We consider a binary classification problem: Given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, $x_i \in R^d$, $y_i \in \{-1, 1\}$. Note that input x_i is called feature vector, output y_i is label. Suppose there is a family of function $f(x, w)$, binary classification aims to specify the best w such that x and y are related.

Based on the input feature vectors and certain type of function we adopt, the output can be predicted based on certain prediction rules. For binary classification problem, the prediction rule can be shown as follows:

$$y = \begin{cases} 1, & f(x,w) \geq 0 \\ -1, & f(x,w) < 0 \end{cases}$$

An error term can be defined as follows to denote whether the output is successfully predicted or not:

$$\text{error} = \begin{cases} 1, & yf(x,w) < 0 \\ 0, & yf(x,w) \geq 0 \end{cases}$$

Obviously our goal is to find the best parameter w in order to minimize the total error in the process of prediction. The optimization step in the binary classification model can be extracted below:

- ◇ empirical risk minimization:

$$\min_w \sum_{i=1}^{n} \mathbb{1}_{\{y_i f(x_i,w)<0\}}$$

- ◇ expected risk minimization:

$$\min_w \mathrm{E}_{(x,y)} \ \mathbb{1}_{\{yf(x,w)<0\}} \Longrightarrow \min_w \mathrm{P}(yf(x,w) < 0)$$

For the purpose of simplicity, we define of 0-1 loss function

$$l(v) = \begin{cases} 1, & v < 0 \\ 0, & v \geq 0 \end{cases}$$

Thus the minimization step in classification model becomes

$$\min_w \sum_{i=1}^{n} l(y_i f_i(x_i, w))$$

which is usually called 0-1 loss minimization problem.

It can be easily seen that 0-1 loss minimization problem is a non-convex optimization problem, which is NP-hard. Instead of solving directly the 0-1 loss minimization problem, one can resort to convex relaxation and solve instead convex upper bound of the 0-1 loss function. We introduce 4 common convex loss functions here:

- hinge loss function: $l(v) = \max\{1 - v, 0\}$
- exponential loss function: $l(v) = \exp(-v)$
- modified least square loss function: $l(v) = \max(1 - v, 0)^2$
- logistic loss function: $l(v) = \log(1 + \exp(-v))$

The comparison between the original non-convex 0-1 loss function and different commonly applied loss functions is shown in Figure 13.1.

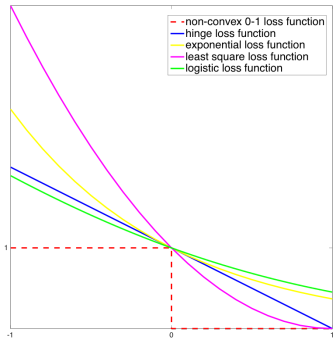


Figure 13.1: 0-1 loss function and common loss functions

13.2.2 Logistic Regression Model

When using the logistic loss as the convex surrogate function, we end up with the logistic regression model

$$\min_w f(w) := \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2}\|w\|_2^2$$

Logistic regression model can also be obtained through MLE by assuming the likelihood

$$P(y = 1|x, w) = \frac{1}{1 + \exp(-w^T x)} = \sigma(w^T x),$$

where $\sigma(u) = \frac{1}{1+\exp(-u)}$ is known as the sigmoid function.

The first order information of regression model is given by

$$\nabla f(w) = \sum_{i=1}^{n} \frac{-y_i x_i \exp(-y_i w^T x_i)}{1 + \exp(-y_i w^T x_i)} + \lambda w = \sum_{i=1}^{n} y_i x_i (\sigma(y_i w^T x_i) - 1) + \lambda w$$

The second order information of regression model is

$$\nabla^2 f(w) = \sum_{i=1}^{n} x_i \sigma(y_i w^T x_i)(1 - \sigma(y_i w^T x_i)) {x_i}^T + \lambda \mathrm{I} = X^T A X + \lambda \mathrm{I}$$

where A is a diagonal matrix with diagonals given by $A_{ii} = \sigma(y_i w^T x_i)(1 - \sigma(y_i w^T x_i)), i = 1, \ldots, n$, $X^T = [x_1, \ldots, x_n]$ is the observation matrix. Note that $0 < A_i i \leq 1/4, \forall i$, hence, we can conclude that

$$L \leq \frac{1}{4}\lambda_{\max}(X^T X) + \lambda.$$

In practice, when L is not known, one can adopt back-tracking (online search) approach to search for the appropriate L until it satisfies the condition $f(x_{t+1}) - f(x_t) \leq \nabla f(x_t)(x_{t+1} - x_t) + \frac{L}{2}\|x_{t+1} - x_t\|^2$. In the following experiments, we will simply use the fixed stepsize $\gamma_t = 1/L$ with the L given above.

13.3 Numerical Implementation

We run Python codes to illustrate how to apply our algorithms to solve logistic regression and investigate their empirical performances.

13.3.1 Data Generation

We use the following codes to generate our inputs x and outputs y in preparation for doing the logistic regression.

```
N = 100
dim = 30
lamda = 1
sigmoid = lambda x: 1/(1+np.exp(-x))
np.random.seed(50)
w = np.matrix(np.random.multivariate_normal([0.0]*dim, np.eye(dim))).T
X = np.matrix(np.random.multivariate_normal([0.0]*dim, np.eye(dim), size = N))
y = 2 * (np.random.uniform(size = (N, 1)) < sigmoid(X*w)) - 1
```

13.3.2 Optimization via CVXPY

Before implementing the algorithm, we apply the CVXPY module solving the optimization problem to specify the optimal solution for sake of conducting comparative investigation later. CVXPY is a Python-embedded modeling language for convex optimization problems based on Interior-Point-Methods. We run the following Python codes to implement CVXPY.

```
import cvxpy as cvx

w = cvx.Variable(dim)
loss = cvx.sum_entries(cvx.logistic(-cvx.mul_elemwise(y, X*w))) + lamda/2 * cvx.sum_squares(w)

problem = cvx.Problem(cvx.Minimize(loss))
problem.solve(verbose=True)
opt = problem.value
```

13.3.3 Gradient-based Algorithms

For all the algorithms below, we initialize $w_0 = 0$.

- Gradient Descent

$$w_{t+1} = w_t - \frac{1}{L}\nabla f(w_t)$$

```
w = np.matrix([0.0]*dim).T
for t in range(0, max_iter):
    obj_val = obj(w)
    w = w - 1/L * grad(w)
```

- Accelerated Gradient Descent

$$w_{t+1} = v_t - \frac{1}{L}\nabla f(v_t)$$
$$v_{t+1} = w_{t+1} + \frac{t}{t+3}(w_{t+1} - w_t)$$

```
w = np.matrix([0.0]*dim).T
v = w
for t in range(0, max_iter):
    obj_val = obj(w)
    w_prev = w
    w = v - 1/L * grad(v)
    v = w + t/(t+3) * (w - w_prev)
```

- FISTA

$$w_{t+1} = v_t - \frac{1}{L}\nabla f(v_t)$$
$$v_{t+1} = w_{t+1} + \frac{a_t - 1}{a_{t+1}}(w_{t+1} - w_t)$$

where $a_0 = 0, \quad a_{t+1} = \frac{1+\sqrt{1+4a_t{}^2}}{2}$.

```
w = np.matrix([0.0]*dim).T
v = w
a = 0.0
for t in range(0, max_iter):
    obj_val = obj(w)
    w_prev = w
    w = v - 1/L*grad(v)
    a_prev = a
    a = (1 + np.sqrt(1 + 4 * a_prev**2))/2
    v = w + (a_prev - 1) / a * (w - w_prev)
```

- Block Coordinate Gradient Descent

$$\text{pick i from } i \in \{1, 2, ..., n\}$$
$$w_{t+1} = w_t - \frac{1}{L}U_i\nabla_i f(w_t)$$

◇ Cyclic Coordinate Gradient Descent

```
w = np.matrix([0.0]*dim).T
for t in range(0, max_iter):
    obj_val = obj(w)
    for i in range(0,dim):
        w[i] = w[i] - 1/L * grad(w)[i]
```

◇ Gauss-Southwell Coordinate Gradient Descent

```
w = np.matrix([0.0]*dim).T
for t in range(0, max_iter):
```

```
    obj_val = obj(w)
    i = np.argmax(abs(grad(w)))
    w[i] = w[i] - 1/L * grad(w)[i]
```

◇ Randomized Coordinate Gradient Descent

```
w = np.matrix([0.0]*dim).T
for t in range(0, max_iter):
    obj_val = obj(w)
    for i in range(0, dim):
        i = np.random.randint(0, dim)
        w[i] = w[i] - 1/L * grad(w)[i]
```

13.3.4 Numerical Comparisons

Next we compare the convergence performance of different algorithms. Particularly we want to examine the effect of strong convexity and choices of stepsizes on the convergence rate of different algorithms. We can set the λ to be 1 to make the regression loss function strongly convex, and set λ to be 0 for the convex case.

Gradient Descent and Accelerated Gradient Descent

First we compare the convergence performance of gradient descent and accelerated gradient descent under strongly convex and convex case respectively. We plot the objective error in each iteration step

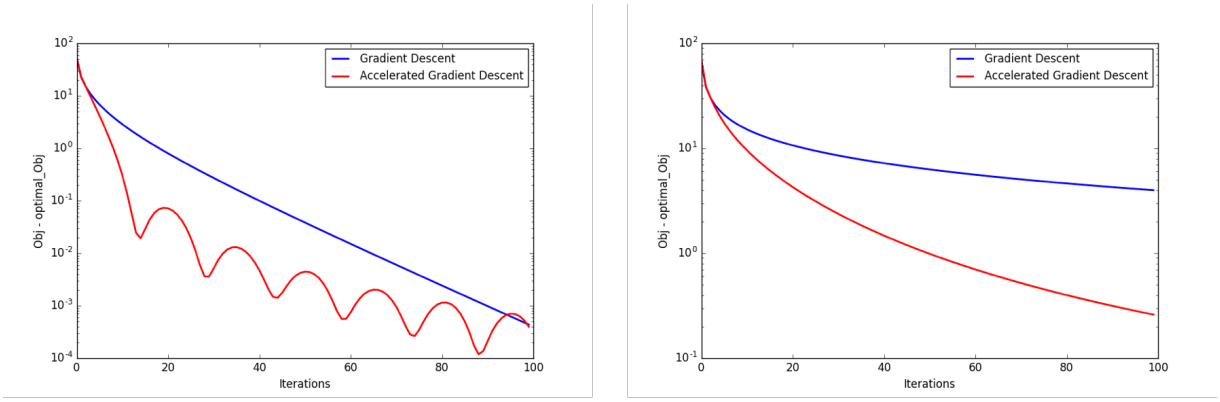


Figure 13.2: Convergence performance of GD and AGD

As you can see from Figure 13.2, accelerated gradient descent (AGD) always achieves a better convergence performance compared with the gradient descent (GD) under both strongly convex case and convex case. It is noticeable that the convergence performance is rather stable and robust in the convex case. In the strongly convex case, convergence error curve end up with drastic oscillation.

Nesterov's Accelerated Gradient Descent and FISTA

We pay attention to two different kinds of accelerated gradient descent methods, i.e., AGD and FISTA. As you can see from Figure 13.3, both AGD and FISTA have pretty similar convergence performance in the convex case and strongly convex case.

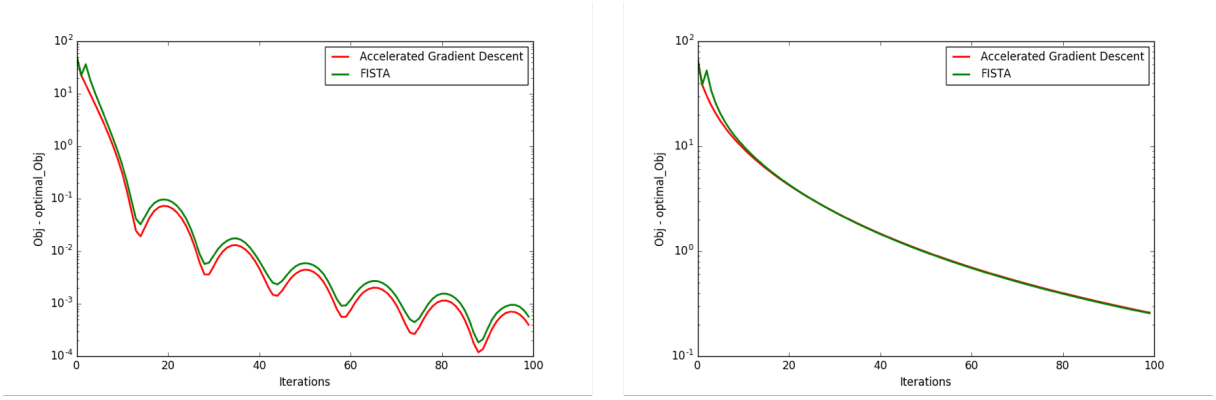


Figure 13.3: Convergence performance of AGD and FISTA

Coordinated Gradient Descent

We also analyze several coordinated gradient descent algorithms with Gauss-Southwell, randomized and cyclic updating rules. We adopt the strongly convex assumption ($\lambda = 1$) and vary the stepsize to explore the convergence performance under small ($\gamma_t = 1/L$) and big stepsizes ($\gamma_t = 1/L_i$).

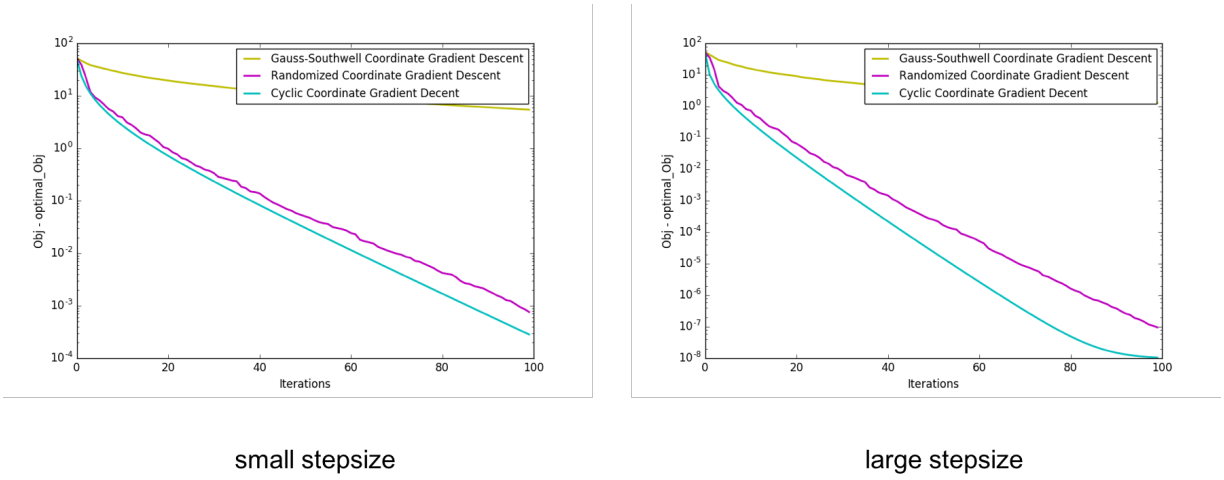


Figure 13.4: Convergence performance of CGD

As Figure 13.4 suggests, Both randomized-CD and cyclic-CD perform far better than Gauss-Southwell-CD, though cyclic-CD outweighs the randomized-CD a bit in both small stepsize case and large stepsize case.

Note that the larger iteration stepsize can lead to more accurate result compared with the small iteration stepsize.

Overall Performance

We have analyzed and compared the performance of 5 different algorithms that can be applied to solve logistic regression problem. Naturally we tend to understand which algorithm behaves the best. Note that our comparison is conducted under the strongly convex case. Like previous case, we consider both a small and large stepsize for the two variants of coordinated gradient descent algorithms.

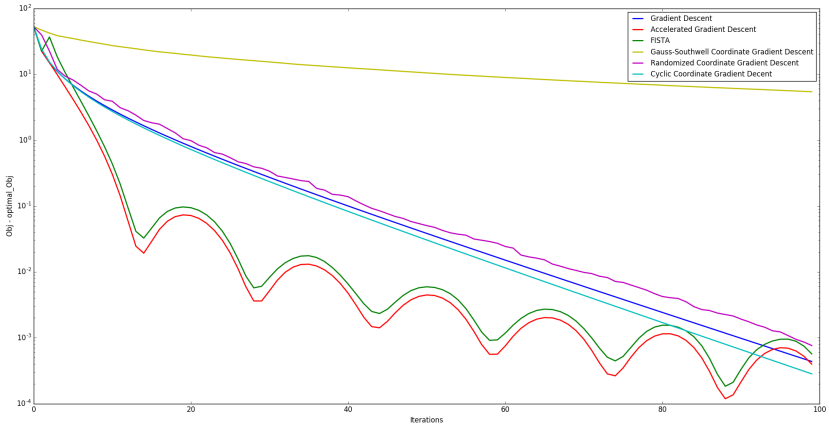


Figure 13.5: Convergence performance summary of 5 different algorithms under small stepsize

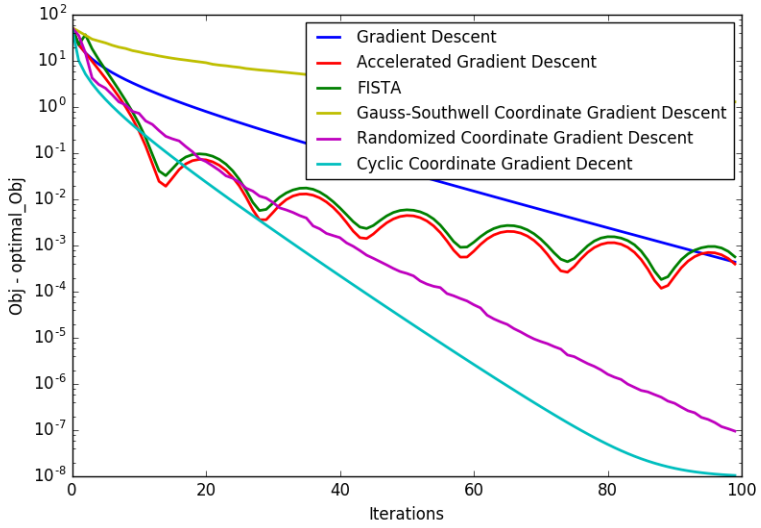


Figure 13.6: Convergence performance summary of 5 different algorithms

Figure 13.5 and 13.6 suggests that when large stepsize is adopted, cyclic-CD and randomized-CD can perform

much better than AGD and FISTA.

There are still several points that haven't been completely addressed in the numerical case study. We see how different stepsize could change drastically on the behavior for CD algorithms. How robust are these algorithms? What happens if we change the stepsize for GD, AGD? Under which regimes would CD be favorable comparing to GD, or cyclic-CD be favorable comparing to randomized-CD? To make a proper conclusion on which algorithm works best among all certainly requires a deeper investigation.

IE 598: Big Data Optimization **Fall 2016**

Lecture 14: Subgradient Method – October 11

Lecturer: Niao He *Scribers: Kaiqing Zhang*

Overview: From this lecture on, we are going to focus on nonsmooth convex optimization problems, where the objective functions are not necessarily continuously differentiable. We study the first algorithm, *subgradient method*, to address such problems. We investigate the convergence rates of the Subgradient Method under various problem settings.

14.1 Problem Setting: Nonsmooth Convex Optimization

In the previous lectures, we mainly discussed smooth convex optimization problems, where the objective function is continuously differentiable and the gradient is Lipschitz continuous. However, such differentiability may not hold for many applications, such as LASSO [Tib96], Support Vector Machine [SuyVan99], or Optimal Control [Vin00]. We now consider nonsmooth convex optimization problems

$$\min_{x\in X} f(x)$$

where

- $X \subset \mathbb{R}^n$ is convex and compact.
- $f : X \to \mathbb{R}$ is convex, possibly non-differentiable, but Lipschitz continuous on X.

Accordingly, we can define two important quantities of X and f as

- $\Omega = \Omega(X) = \max\limits_{x,y\in X} \frac{1}{2}\|x-y\|_2^2$ is the diameter of X.
- $M = M_{\|\cdot\|_2}(f) = \sup\limits_{x,y\in X} \frac{|f(x)-f(y)|}{\|x-y\|_2} < +\infty$ is the constant that characterizes the Lipschitz continuity.

Note that as discussed in Lecture 4, the convexity of the function f can actually lead to local Lipschitz continuity. We will simply make this an assumption in the in the subsequent text. In addition, due to the convexity, subgradient of f always exists, which motivates the Subgradient Method as follows.

14.2 Subgradient Method [N. Z. Shor, 1967]

The subgradient method, also called *subgradient descent*, was first proposed by Dr. Naum Zuselevich Shor in 1967. It works as follows

Algorithm 1 Subgradient Method

1: Pick $x_1 \in X$
2: **while** the iteration converges **do**
3: Pick $g(x_t) \in \partial f(x_t)$
4: $x_{t+1} = \Pi_X(x_t - \gamma_t g(x_t))$

where

- $g(x_t) \in \partial f(x_t)$ is a subgradient of f at x_t, which implies that $f(y) \geq f(x_t) + g(x_t)^T(y - x_t), \forall y \in X$.
- $\gamma_t > 0$ is the stepsize.
- $\Pi_X(x) = \arg\min_{y \in X} \|x - y\|_2^2$ is the projection operation.

Remark 14.1 *Unlike Gradient Descent (GD), Subgradient Descent (SD) method is not a descent method, i.e. moving along negative direction subgradient is not necessarily decreasing the objective function.*

To illustrate this, consider the function $f(x) = |x_1| + 2|x_2|$ with the level set contour as in Figure 15.1. For

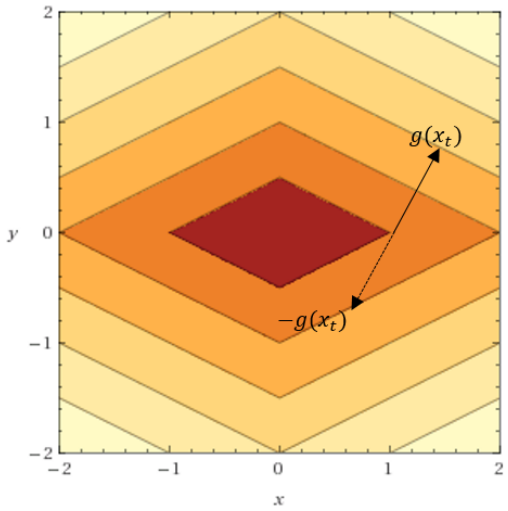


Figure 14.1: The contour of the function $f(x) = |x_1| + 2|x_2|$ and one of the subgradient direction at $(1, 0)$.

example, at $x = (1, 0)$, the the subdifferentiable set is $\partial f(x) = \{(1, 2a), a \in [-1, 1]\}$. If we choose $g = (1, 2)$ as shown, $-g$ is obviously not a descent direction.

Choices of Stepsizes Stepsize γ_t is an important parameter that need to be selected during the iterations, which will affect the convergence analysis as we will show later. Four most commonly used stepsizes include:

1. Constant stepsize: $\gamma_t \equiv \gamma > 0$.
2. Scaled stepsize: $\gamma_t = \frac{\gamma}{\|g(x_t)\|_2}$.
3. Non-summable but diminishing stepsize satisfying:

$$\sum_{t=1}^{\infty} \gamma_t = \infty, \qquad \lim_{t \to \infty} \gamma_t = 0$$

   e.g., $\gamma_t \sim O(\frac{1}{\sqrt{t}})$.

4. Non-summable but square-summable stepsize satisfying:

$$\sum_{t=1}^{\infty} \gamma_t = \infty, \qquad \sum_{t=1}^{\infty} \gamma_t^2 < \infty,$$

   e.g., $\gamma_t \sim O(\frac{1}{t})$.

5. Polyak stepsize: Assuming $f_* = f(x_*)$ is known, choose

$$\gamma_t = \frac{f(x_t) - f_*}{\|g(x_t)\|_2^2}.$$

14.3 Convergence Analysis

To be discussed later, it turns out that subgradient descent behaves substantially different from gradient descent. The choices of stepsize, rates of convergence, and criterion used to measure the convergence are different. As mentioned in Remark 14.1, subgradient descent is not a descent method. Hence we will need to introduce other quantities to measure the convergence, instead of the quantity $f(x_t) - f_*$ as used earlier.

14.3.1 Convex Case

For convex f, we have the following theorem.

Theorem 14.2 *Assume f is convex, then subgradient method satisfies*

$$\min_{1 \le t \le T} f(x_t) - f_* \le \left(\sum_{t=1}^{T} \gamma_t \right)^{-1} \left(\frac{1}{2} \|x_1 - x_*\|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \gamma_t^2 \|g(x_t)\|_2^2 \right) \tag{14.1}$$

and

$$f(\hat{x}_T) - f_* \le \left(\sum_{t=1}^{T} \gamma_t \right)^{-1} \left(\frac{1}{2} \|x_1 - x_*\|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \gamma_t^2 \|g(x_t)\|_2^2 \right) \tag{14.2}$$

where $\hat{x}_T = \left(\sum_{t=1}^{T} \gamma_t \right)^{-1} \left(\sum_{t=1}^{T} \gamma_t x_t \right) \in X$.

Proof: The proof uses the similar technique as in the convergence for smooth problems. First

$$\begin{aligned} \|x_{t+1} - x_*\|_2^2 =& \|\Pi_X(x_t - \gamma_t g(x_t)) - \Pi_X(x_*)\|_2^2 \\ \le& \|x_t - \gamma_t g(x_t) - x_*\|_2^2 \\ =& \|x_t - x_*\|_2^2 - 2\gamma_t g(x_t)^T (x_t - x_*) + \gamma_t^2 \|g(x_t)\|_2^2 \end{aligned}$$

the inequality comes from the non-expansiveness of the projection operation. Therefore we have

$$\gamma_t g(x_t)^T (x_t - x_*) \le \frac{1}{2} \left(\|x_t - x_*\|_2^2 - \|x_{t+1} - x_*\|_2^2 + \gamma_t^2 \|g(x_t)\|_2^2 \right). \tag{14.3}$$

Due to the convexity of f, we have

$$\gamma_t g(x_t)^T (x_t - x_*) \geq \gamma_t \left(f(x_t) - f_*\right). \tag{14.4}$$

Combining (14.3) and (14.4) and adding both sides of the inequality from $t = 1$ to $t = T$, we obtain

$$\begin{aligned}\sum_{t=1}^{T} \gamma_t \left(f(x_t) - f_*\right) \leq & \frac{1}{2}\left(\|x_1 - x_*\|_2^2 - \|x_{T+1} - x_*\|_2^2 + \sum_{t=1}^{T} \gamma_t^2 \|g(x_t)\|_2^2\right) \\ \leq & \frac{1}{2}\left(\|x_1 - x_*\|_2^2 + \sum_{t=1}^{T} \gamma_t^2 \|g(x_t)\|_2^2\right). \end{aligned} \tag{14.5}$$

For the proof of (14.1), by definition, the left hand side of (14.5) can be lower bounded by

$$\sum_{t=1}^{T} \gamma_t \left(f(x_t) - f_*\right) \geq \left(\sum_{t=1}^{T} \gamma_t\right) \cdot \left(\min_{1 \leq t \leq T} f(x_t) - f_*\right)$$

For the proof of (14.2), first note that $\hat{x}_T$ is a convex combination of $\{x_1, \cdots, x_T\}$. Due to the convexity of f, we have

$$\sum_{t=1}^{T} \gamma_t f(x_t) \geq \left(\sum_{t=1}^{T} \gamma_t\right) \cdot f(\hat{x}_T)$$

and left hand side of (14.5) is thus lower bounded by

$$\sum_{t=1}^{T} \gamma_t \left(f(x_t) - f_*\right) \geq \left(\sum_{t=1}^{T} \gamma_t\right) \cdot \left(f(\hat{x}_T) - f_*\right).$$

Bounds (14.1) and (14.2) are hence proved. ■

Remark. Invoking the definition of Ω and M, we have $\frac{1}{2}\|x_1 - x_*\|_2^2 \leq \Omega$ and $\|g(x_t)\|_2 \leq M$. As a corollary,

$$\begin{aligned}\min_{1 \leq t \leq T} f(x_t) - f_* &\leq \left(\Omega + \frac{1}{2}\sum_{t=1}^{T} \gamma_t^2 M^2\right) \Bigg/ \left(\sum_{t=1}^{T} \gamma_t\right), \\ f\left(\frac{\sum_{t=1}^{T} \gamma_t x_t}{\sum_{t=1}^{T} \gamma_t}\right) - f_* &\leq \left(\Omega + \frac{1}{2}\sum_{t=1}^{T} \gamma_t^2 M^2\right) \Bigg/ \left(\sum_{t=1}^{T} \gamma_t\right).\end{aligned}$$

Remark 2. Slightly modifying the summation or averaging from T_0 to T instead of from 1 to T, we end up with a even more general results

$$\begin{aligned}\min_{T_0 \leq t \leq T} f(x_t) - f_* &\leq \left(\Omega + \frac{1}{2}\sum_{t=T_0}^{T} \gamma_t^2 M^2\right) \Bigg/ \left(\sum_{t=T_0}^{T} \gamma_t\right), \quad \forall 1 \leq T_0 \leq T \\ f\left(\frac{\sum_{t=T_0}^{T} \gamma_t x_t}{\sum_{t=T_0}^{T} \gamma_t}\right) - f_* &\leq \left(\Omega + \frac{1}{2}\sum_{t=T_0}^{T} \gamma_t^2 M^2\right) \Bigg/ \left(\sum_{t=T_0}^{T} \gamma_t\right), \quad \forall 1 \leq T_0 \leq T.\end{aligned}$$

Convergence with various stepsizes. It is interesting to see how the bounds in (14.1) and (14.1) would imply the convergence and even the convergence rate with different choices of stepsizes. By abuse of notation, we denote both $\min_{1\le t\le T} f(x_t) - f_*$ and $f(\hat{x}_T) - f_*$ as ϵ_T.

1. *Constant stepsize:* with $\gamma_t \equiv \gamma$,

$$\epsilon_T \le \frac{\Omega + (T/2)\gamma^2 M^2}{T\gamma} = \frac{\Omega}{T}\cdot\frac{1}{\gamma} + \frac{M^2}{2}\gamma \xrightarrow{T\to\infty} \frac{M^2}{2}\gamma.$$

   It is worth noticing that the error upper-bound does not diminish to zero as T grows to infinity, which shows one of the drawbacks of using arbitrary constant stepsizes. In addition, to optimize the upper bound, we can select the optimal stepsize γ_* to obtain:

$$\gamma_* = \frac{\sqrt{2\Omega}}{M\sqrt{T}} \Rightarrow \epsilon_T \le \frac{\sqrt{2\Omega}M}{\sqrt{T}}.$$

   It is shown that under this optimal choice $\epsilon_T \sim O(\frac{\sqrt{\Omega}M}{\sqrt{T}})$. However, this exhibits another drawback of constant stepsize that in practice T is not known in prior for evaluating the optimal γ_*.

2. *Scaled stepsize:* with $\gamma_t = \frac{\gamma}{\|g(x_t)\|_2}$,

$$\epsilon_T \le \frac{\Omega + (1/2)\gamma^2 T}{\gamma\sum_{t=1}^{T} 1/\|g(x_t)\|_2} \le M\left(\frac{\Omega}{T}\cdot\frac{1}{\gamma} + \frac{1}{2}\gamma\right) \xrightarrow{T\to\infty} \frac{M}{2}\gamma.$$

   Similarly, we can select the optimal γ by minimizing the right hand side, i.e. $\gamma_* = \frac{\sqrt{2\Omega}}{\sqrt{T}}$:

$$\gamma_t = \frac{\sqrt{2\Omega}}{\sqrt{T}\|g(x_t)\|_2} \Rightarrow \epsilon_T \le \frac{\sqrt{2\Omega}M}{\sqrt{T}}.$$

   The same convergence rate is achieved while the same drawback about not knowing T in prior still exists in choosing γ_t.

3. *Non-summable but diminishing stepsize:*

$$\begin{aligned}\epsilon_T &\le \left(\Omega + \frac{1}{2}\sum_{t=1}^{T}\gamma_t^2 M^2\right) \Big/ \left(\sum_{t=1}^{T}\gamma_t\right) \\ &\le \left(\Omega + \frac{1}{2}\sum_{t=1}^{T_1}\gamma_t^2 M^2\right) \Big/ \left(\sum_{t=1}^{T}\gamma_t\right) + \left(\frac{M^2}{2}\sum_{t=T_1+1}^{T}\gamma_t^2\right) \Big/ \left(\sum_{t=T_1+1}^{T}\gamma_t\right)\end{aligned}$$

   where $1 \le T_1 \le T$. When $T \to \infty$, select large T_1 and the first term on the right hand side $\to 0$ since γ_t is non-summable. The second term also $\to 0$ because γ_t^2 always approaches zero faster than γ_t. Consequently, we know that

$$\epsilon_T \xrightarrow{T\to\infty} 0.$$

   An example choice of the stepsize is $\gamma_t = O\left(\frac{1}{t^q}\right)$ with $q \in (0, 1]$. As in the above cases, if we choose $\gamma_t = \frac{\sqrt{2\Omega}}{M\sqrt{t}}$, then

$$\gamma_t = \frac{\sqrt{2\Omega}}{M\sqrt{t}} \Rightarrow \epsilon_T \le O\left(\frac{\sqrt{\Omega}M\ln(T)}{\sqrt{T}}\right).$$

In fact, if we choose the averaging from $\frac{T}{2}$ instead of 1, we have

$$\min_{T/2\le t\le T} f(x_t) - f_* \le O\left(\frac{M\cdot\sqrt{\Omega}}{\sqrt{T}}\right).$$

4. *Non-summable but square-summable stepsize:* It is obvious that

$$\epsilon_T \le \left(\Omega + \frac{M^2}{2}\sum_{t=1}^{T}\gamma_t^2\right) \Bigg/ \left(\sum_{t=1}^{T}\gamma_t\right) \xrightarrow{T\to\infty} 0.$$

A typical choice of $\gamma_t = \frac{1}{t^{1+q}}, q > 0$ also result in the rate of $O(\frac{1}{\sqrt{T}})$.

5. *Polyak stepsize:* The motivation of choosing this stepsize comes from the fact that

$$\begin{aligned}\|x_{t+1} - x_*\|_2^2 \le& \|x_t - x_*\|_2^2 - 2\gamma_t g(x_t)^T(x_t - x_*) + \gamma_t^2\|g(x_t)\|_2^2\\ \le& \|x_t - x_*\|_2^2 - 2\gamma_t\left(f(x_t) - f_*\right) + \gamma_t^2\|g(x_t)\|_2^2\end{aligned}$$

as we showed in the proof of **Theorem** 14.2. The Polyak step $\gamma_t = \frac{f(x_t)-f_*}{\|g(x_t)\|_2^2}$ is exactly the minimizer of the right hand side. In fact, the stepsize yields

$$\|x_{t+1} - x_*\|_2^2 \le \|x_t - x_*\|_2^2 - \frac{\left(f(x_t) - f_*\right)^2}{\|g(x_t)\|_2^2}, \tag{14.6}$$

which guarantees $\|x_t - x_*\|_2^2$ decreases each step. Applying (14.6) recursively, we obtain

$$\sum_{t=1}^{T}\left(f(x_t) - f_*\right)^2 \le 2\Omega\cdot M < \infty.$$

Therefore we have $\epsilon_T \to 0$ as $T\to\infty$ and $\epsilon_T \le O\left(\frac{1}{\sqrt{T}}\right)$ (otherwise ϵ_T will not be square-summable).

14.3.2 Strongly Convex Case

For strongly convex function f, we obtain the following theorem.

Theorem 14.3 *Assume f is μ-strongly convex, then subgradient method with stepsize $\gamma_t = \frac{1}{\mu t}$ satisfies*

$$\min_{1\le t\le T} f(x_t) - f_* \le \frac{M^2(\ln(T)+1)}{2\mu T} \tag{14.7}$$

and

$$f\left(\hat{x}_T\right) - f_* \le \frac{M^2(\ln(T)+1)}{2\mu T}, \qquad \text{with } \hat{x}_T := \frac{1}{T}\sum_{t=1}^{T} x_t \tag{14.8}$$

where M is as defined in Section 14.1.

Proof: First recall that μ-strongly convex implies that

$$f(y) \ge f(x) + g(x)^T(y-x) + \frac{\mu}{2}\|x-y\|_2^2, \quad \forall x, y\in X.$$

Similarly as the proof for **Theorem** 14.2, the left hand side of (14.3) can be lower bounded by

$$\gamma_t g(x_t)^T(x_t - x_*) \geq \gamma_t \left(f(x_t) - f_* + \frac{\mu}{2}\|x_t - x_*\|_2^2 \right).$$

Combining (14.3) and plug in $\gamma_t = \frac{1}{\mu t}$ we have

$$f(x_t) - f_* \leq \left(\frac{\mu}{2}(t-1)\|x_t - x_*\|_2^2 - \frac{\mu}{2}t\|x_{t+1} - x_*\|_2^2 + \frac{1}{2\mu t}\|g(x_t)\|_2^2 \right).$$

By recursively adding both sides from $t = 1$ to $t = T$, we obtain

$$\sum_{t=1}^{T} f(x_t) - f_* \leq \sum_{t=1}^{T} \frac{1}{2\mu t}\|g(x_t)\|_2^2 \leq \frac{M^2}{2\mu}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{M^2}{2\mu}(\ln(T) + 1).$$

In addition, we have $\sum_{t=1}^{T} f(x_t) - f_* \geq T \cdot \epsilon_T$ for either $\min_{1\leq t\leq T} f(x_t) - f_*$ or $f(\hat{x}_T) - f_*$ with $\hat{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ as shown in the previous proof, which leads to the bounds (14.7) and (14.8). ■

With another choice of stepsize and averaging strategy, we can get rid of the log factor in the bound. The following theorem can be obtained [Bub14].

Theorem 14.4 *Assume f is μ-strongly convex, then subgradient method with stepsize $\gamma_t = \frac{2}{\mu(t+1)}$ satisfies*

$$\min_{1\leq t\leq T} f(x_t) - f_* \leq \frac{2M^2}{\mu(T+1)} \tag{14.9}$$

and

$$f(\hat{x}_T) - f_* \leq \frac{2M^2}{\mu(T+1)}, \quad \text{with } \hat{x}_T = \sum_{t=1}^{T}\frac{2t}{T(T+1)}x_t. \tag{14.10}$$

, M is as defined in Section 15.1.

Proof: Similarly as the proof for **Theorem** 14.3, we first have

$$\gamma_t g(x_t)^T(x_t - x_*) \geq \gamma_t \left(f(x_t) - f_* + \frac{\mu}{2}\|x_t - x_*\|_2^2 \right).$$

Substitute $\gamma_t = \frac{2}{\mu(t+1)}$ in 14.3, we have

$$f(x_t) - f_* \leq \left(\frac{\mu}{4}(t-1)\|x_t - x_*\|_2^2 - \frac{\mu}{4}(t+1)\|x_{t+1} - x_*\|_2^2 + \frac{1}{\mu(t+1)}\|g(x_t)\|_2^2 \right).$$

Multiplying both sides by t leads to

$$\begin{aligned} t\left(f(x_t) - f_*\right) &\leq \left(\frac{\mu}{4}t(t-1)\|x_t - x_*\|_2^2 - \frac{\mu}{4}t(t+1)\|x_{t+1} - x_*\|_2^2 + \frac{t}{\mu(t+1)}\|g(x_t)\|_2^2 \right) \\ &\leq \left(\frac{\mu}{4}t(t-1)\|x_t - x_*\|_2^2 - \frac{\mu}{4}t(t+1)\|x_{t+1} - x_*\|_2^2 + \frac{1}{\mu}\|g(x_t)\|_2^2 \right). \end{aligned}$$

Recursively adding both sides from $t = 1$ to $t = T$, we obtain

$$\sum_{t=1}^{T} t\left(f(x_t) - f_*\right) \leq \frac{T}{\mu}\|g(x_t)\|_2^2 \leq \frac{T}{\mu}M^2. \tag{14.11}$$

Moreover, by definition and convexity we have

$$\sum_{t=1}^{T} tf(x_t) \geq \left(\frac{T(T+1)}{2}\right) \cdot f\left(\sum_{t=1}^{T} \frac{2t}{T(T+1)}x_t\right),$$

and

$$\sum_{t=1}^{T} tf(x_t) \geq \left(\frac{T(T+1)}{2}\right) \cdot \min_{1\leq t\leq T} f(x_t).$$

Combining with (14.11) and dividing both sides by $\frac{T(T+1)}{2}$, we conclude the proof. ■

14.4 Summary

It is worth comparing the convergence rate of subgradient method for nonsmooth convex optimization problems with the optimal rate for smooth problems. Actually, it is shown in Table 14.1 that in both convex and strongly convex cases, subgradient method achieves slower convergence than the accelerated gradient descent method. Particularly, subgradient method can only achieve sublinear convergence even under the strongly convex case, instead of linear rate in smooth case. By constructing worst case functions, we will show that this is actually the optimal rate one can get for nonsmooth convex optimization next lecture.

Table 14.1: Comparison of convergence rates for nonsmooth and smooth convex optimization problems.

	Convex	**Strongly Convex**
Subgradient method	$O\left(\frac{\sqrt{\Omega}M}{\sqrt{t}}\right)$	$O\left(\frac{M^2}{\mu t}\right)$
Accelerated gradient descent	$O\left(\frac{L\cdot D^2}{t^2}\right)$	$O\left(\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^{2t}\right)$

References

[Tib96] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[SuyVan99] J. Suykens, and J. Vandewalle, "Least squares support vector machine classifiers", *Neural processing letters*, 9(3), 293-300, 1999.

[Vin00] R. Vinter, "Optimal control. Systems & control: foundations & applications", *Birkauser, Boston*, 2000.

[Nes13] Y. Nesterov, "Introductory lectures on convex optimization: A basic course", *Springer Science & Business Media*, vol. 87, 2013.

[Bub14] S. Bubeck, "Convex optimization: Algorithms and complexity", arXiv preprint arXiv:1405.4980, 2014.

IE 598: Big Data Optimization **Fall 2016**

Lecture 15: From Subgradient Decent to Mirror Decent – October 13

Lecturer: Niao He *Scriber: Hongyi LI*

15.1 Recap

Recall that our goal is to solve the nonsmooth convex problem

$$\min_{x\in X} f(x)$$

where the objective function f is (possibly non-differentiable) convex, Lipschitz continuous, and the set X is convex and compact. We use the following notations to describe the Lipschitz continuity and diameter of the set:

$$\begin{cases} M = M_{\|.\|_2}(f) := \sup_{x\in X} \frac{|f(x)-f(y)|}{\|x-y\|_2} \\ \Omega = \Omega(X) := \max_{x,y\in X} \frac{1}{2}\|x-y\|_2^2 \end{cases}$$

Subgradient Decent: at each iteration, runs a projection along the negative subgradient direction:

$$x_{t+1} = \Pi_X(x_t - \gamma_t g(x_t)) \tag{15.1}$$

Main results:

$$\min_{1\le t\le T} f(x_t) - f_* \le \begin{cases} O(\frac{\sqrt{\Omega}M}{\sqrt{T}}) & \text{if } f \text{ is convex;} \\ O(\frac{M^2}{\mu T}) & \text{if } f \text{ is } \mu\text{-strongly convex} \end{cases}$$

Key Inequality (used to show the results):

$$\gamma_t g(x_t)^T(x_t - x_*) \le \frac{1}{2}\|x_t - x_*\|_2^2 - \frac{1}{2}\|x_{t+1} - x_*\|_2^2 + \frac{\gamma_t^2}{2}\|g(x_t)\|_2^2 \tag{15.2}$$

15.2 Lower Bounds for Non-smooth Convex Optimization

While the convergence rates achieved by subgradient descent seems much worse than those achieved by gradient descent for smooth problems, we show below that in the worst case, one cannot improve the $O(1/\sqrt{t})$ and $O(1/t)$ rates for the convex and strongly convex situations, respectively, when using only block-box oriented methods that only have access to the subgradient of the objective function. The worst case function is given by the piecewise-linear function $f(x) = \max_{1\le i\le n} x_i$. We provide details below.

Theorem 15.1 *For any $1 \le t \le n$, $x_1 \in \mathbb{R}_n$,*
(1) there exists a M-Lipschitz continuous function f and a convex set X with diameter Ω, such that for any first-order method that generates:

$$x_t \in x_1 + span(g(x_1), ..., g(x_{t-1})), \text{ where } g(x_i) \in \partial f(x_i), i = 1, \dots, t-1$$

we always have

$$\min_{1\le s\le t} f(x_s) - f_* \ge \frac{\sqrt{\Omega}M}{2\sqrt{2}(1+\sqrt{t})}$$

(2) there exists a μ-strongly convex, M-Lipschitz continuous function f and a convex set X with diameter Ω, for any first-order method as described above, we always have

$$\min_{1\le s\le t} f(x_s) - f_* \ge \frac{M^2}{8\mu t}$$

Proof: (Nemirovski & Yudin 1979)
Let $X = \{x \in \mathbb{R}^n, \|x\|_2 \le R\}$ where $R = \sqrt{\frac{\Omega}{2}}$. Then $\frac{1}{2}\|x-y\|_2^2 \le \|x\|_2^2 + \|y\|_2^2 \le 2R^2$. Hence, $\Omega(X) \le \Omega$.
Let $f : \mathbb{R}^n \to \mathbb{R}$ s.t.

$$f(x) = C \cdot \max_{1\le i\le t} x_i + \frac{\mu}{2}\|x\|_2^2,$$

where C is some constant to be determined.

The subgradient of function f is given by,

$$\partial f(x) = \mu x + C \cdot conv\{e_i : i \text{ such that } x_i = \max_{1\le j\le t} x_j\}$$

Note that the optimal solution and optimal value of the porblem $\min_{x\in X} f(x)$ is given by

$$x_{*i} = \begin{cases} -\frac{C}{\mu t} & 1 \le i \le t \\ 0 & t < i \le n \end{cases} \quad \text{and } f_* = -\frac{C^2}{2\mu t}.$$

This is because, for $t < i \le n$, x_{*i}'s only increase $\frac{\mu}{2}\|x\|_2^2$, we can just set them all to 0; for $1 \le i \le t$, f is symmetric for all x_{*i}.

Without loss of generality, set $x_1 = 0$ and consider the worst subgradient oracle, that given an input x, it returns $g(x) = C \cdot e_i + \mu x$, with i being the first coordinate that $x_i = \max_{1\le j\le t} x_j$.

By induction, we can show that $x_t \in span(e_1, ..., e_{t-1})$. This implies for $1 \le s \le t$, $f(x_s) \ge 0$. Therefore

$$\min_{1\le s\le t} f(x_s) - f_* \ge \frac{C^2}{2\mu t}.$$

(1) Let $C = \frac{M\sqrt{t}}{1+\sqrt{t}}$, $\mu = \frac{M}{R(1+\sqrt{t})}$, then $\|\partial f(x)\|_2 \le C + \mu\|x\|_2 \le C + \mu R = M$. This implies that $f(x)$ is M-Lipschitz continuous. Moreover, we have

$$\min_{1\le s\le t} f(x_s) - f_* \ge \frac{C^2}{2\mu t} = \frac{M\sqrt{\Omega}}{2\sqrt{2}(1+\sqrt{t})}$$

(2) Let $C = \frac{M}{2}$, $\mu = \frac{M}{2R}$, then $\|\partial f(x)\|_2 \le C + \mu R = M$. This implies that $f(x)$ is M-Lipschitz continuous, μ-strongly convex.

$$\min_{1\le s\le t} f(x_s) - f_* \ge \frac{C^2}{2\mu t} = \frac{M^2}{8\mu t}$$

If the method returns $g(x)$ not be e_i where i is the smallest coordinate such that $x_i = \max_{1\le j\le t} x_j$, then we have $x_t \in span(e_{i_1}, ..., e_{i_{t-1}})$ for some $i_1, \ldots, i_{t-1}$, the situation is only getting worse; we still have $f(x_s) \ge 0, \forall s \le t$ and the analysis still hold. ■

Remark: To obtain an ϵ-solution, the number of subgradient call is $O(\frac{M^2\Omega}{\epsilon^2})$ for convex function, and $O(\frac{M^2}{\mu\epsilon})$ for strongly convex function. The above theorem indicates that these complexity bounds are indeed optimal.

Let us take a close look at the complexity bound for convex function,

$$O(\frac{M^2\Omega}{\epsilon^2}) = O\left(\frac{[M_{\|.\|_2}(f)\cdot \max_{x,y\in X}\|x-y\|_2]^2}{\epsilon^2}\right)$$

where the term $M_{\|.\|_2}(f)\cdot \max_{x,y\in X}\|x-y\|_2$ can be considered as the $\|.\|_2$-variation of f on $x\in X$.

15.3 Mirror Decent

Recall the subgradient decent updating rule can be equivalently written as

$$x_{t+1} = \operatorname*{argmin}_{x\in X}\{\frac{1}{2}\|x-x_t\|_2^2 + \langle\gamma_t g(x_t), x\rangle\} = \operatorname*{argmin}_{x\in X}\{\frac{1}{2}\|x\|_2^2 + \langle\gamma_t g(x_t) - x_t, x\rangle\}$$

Why should we use the Euclidean $\|.\|_2$ distance?
We will introduce a new algorithm, *Mirror Descent*, that generalizes subgradient descent with non-Euclidean distances.

15.3.1 Basic Setup

The Mirror Descent algorithm works as follows

$$x_{t+1} = \operatorname*{argmin}_{x\in X}\{V_\omega(x,x_t) + \langle\gamma_t g(x_t), x\rangle\} = \operatorname*{argmin}_{x\in X}\{\omega(x) + \langle\gamma_t g(x_t) - \nabla\omega(x_t), x\rangle\}$$

where

- **Bregman distance:** $V_\omega(x,y) = \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$;
- **Distance generating function:** $\omega(x): X\to\mathbb{R}$ should be convex, continuously differentiable and 1-strongly convex with respect to some norm $\|.\|$, i.e. $\omega(x)\geq\omega(y)+\nabla\omega(y)^T(x-y)+\frac{1}{2}\|x-y\|^2$.

Note: Bregman distance is not a valid distance: $V_\omega(x,y)\neq V_\omega(y,x)$ and triangle inequality may not hold. This is often referred to as Bregman divergence. By definition, we always have $V_\omega(x,y)\geq\frac{1}{2}\|x-y\|^2$.

Given an input x and vector ξ, we will define the **prox mapping**:

$$\text{Prox}_x(\xi) = \operatorname*{argmin}_{u\in X}\{V_\omega(u,x) + \langle\xi,u\rangle\} \tag{15.3}$$

Mirror Descent update can be simplified as

$$x_{t+1} = \text{Prox}_{x_t}(\gamma_t g(x_t))$$

Example 1 (ℓ_2 setup): $X\subseteq\mathbb{R}^n$, $\omega(x)=\frac{1}{2}\|x\|_2^2$, $\|x\|=\|x\|_2$, then

(a) Bregman distance redues to $V_\omega(x,y)=\frac{1}{2}\|x-y\|_2^2$

(b) Prox-mapping reduces to $\text{Prox}_x(\xi)=\Pi_X(x-\xi)$

(c) Mirror decent reduces to subgradient decent.

Example 2 (ℓ_1 setup) : $X = \{x \in \mathbb{R}^n_+, \sum_{i=1}^n x_i = 1\}$, $\omega(x) = \sum_{i=1}^n x_i \ln(x_i)$, $\|x\| = \|x\|_1$. One can verify that $\omega(x)$ is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on X. In this case, we have

(a) Bregman distance becomes to $V_\omega(x, y) = \sum_{i=1}^n x_i \ln(x_i/y_i)$, known as the Kullback-Leibler divergence.

(b) Prox-mapping becomes to

$$\text{Prox}_x(\xi) = \left(\sum_{i=1}^n x_i e^{-\xi_i}\right)^{-1} \begin{bmatrix} x_1 e^{-\xi_1} \\ \dots \\ x_n e^{-\xi_n} \end{bmatrix}$$

(c) Mirror decent gives rise to multiplicative updates with normalization.

15.3.2 Convergence of Mirror ecent

We first present the useful three point identity lemma:

Lemma 15.2 (Three point identity) *For any $x, y, z \in dom(\omega)$:*

$$V_\omega(x, z) = V_\omega(x, y) + V_\omega(y, z) - \langle \nabla\omega(z) - \nabla\omega(y), x - y\rangle$$

Proof: This can be easily derived from definiton. We have

$$\begin{aligned} V_\omega(x, y) + V_\omega(y, z) &= \omega(x) - \omega(y) + \omega(y) - \omega(z) - \langle \nabla\omega(y), x - y\rangle - \langle \nabla\omega(z), y - z\rangle \\ &= V_\omega(x, z) + \langle \nabla\omega(z), x - z\rangle - \langle \nabla\omega(y), x - y\rangle - \langle \nabla\omega(z), y - z\rangle \\ &= V_\omega(x, z) + \langle \nabla\omega(z) - \nabla\omega(y), x - y\rangle \end{aligned}$$

■

Note: when $\omega(x) = \frac{1}{2}\|w\|_2^2$, this is the same as Law of cosines i.e.

$$\|z - x\|_2^2 = \|z - y\|_2^2 + \|y - x\|_2^2 + 2\langle z - y, y - x\rangle.$$

Theorem 15.3 *For mirror decent, let f be convex, then we have:*

$$\min_{1\le t\le T} f(x_t) - f_* \le \frac{V_\omega(x_*, x_1) + \frac{1}{2}\sum_{t=1}^T \gamma_t^2 \|g(x_t)\|_*^2}{\sum_{t=1}^T \gamma_t} \tag{15.4}$$

and

$$f\left(\frac{\sum_{t=1}^T \gamma_t x_t}{\sum_{t=1}^T \gamma_t}\right) - f_* \le \frac{V_\omega(x_*, x_1) + \frac{1}{2}\sum_{t=1}^T \gamma_t^2 \|g(x_t)\|_*^2}{\sum_{t=1}^T \gamma_t} \tag{15.5}$$

where $\|.\|_*$ denotes dual norm.

Proof: Since $x_{t+1} = \text{argmin}_{x\in X}\{\omega(x) + \langle \gamma_t g(x_t) - \nabla\omega(x_t), x\rangle\}$, by optimality condition, we have

$$\langle \nabla\omega(x_{t+1}) + \gamma_t g(x_t) - \nabla\omega(x_t), x - x_{t+1}\rangle \ge 0$$

From Three point identity, we have for $\forall x \in X$:

$$\langle \gamma_t g(x_t), x_{t+1} - x\rangle \le \langle \nabla\omega(x_{t+1}) - \nabla\omega(x_t), x - x_{t+1}\rangle = V_\omega(x, x_t) - V_\omega(x, x_{t+1}) - V_\omega(x_{t+1}, x_t)$$

$$\langle \gamma_t g(x_t), x_t - x\rangle \le V_\omega(x, x_t) - V_\omega(x, x_{t+1}) - V_\omega(x_{t+1}, x_t) + \langle \gamma_t g(x_t), x_t - x_{t+1}\rangle$$

By Young's inequality,

$$\langle \gamma_t g(x_t), x_t - x_{t+1}\rangle \le \frac{\gamma_t^2}{2}\|g(x_t)\|_*^2 + \frac{1}{2}\|x_t - x_{t+1}\|^2.$$

From the strongly convexity of $\omega(x)$, $V_\omega(x_{t+1}, x_t) \ge \frac{1}{2}\|x_t - x_{t+1}\|^2$. Adding these two inequalities, we get the key inequality:

$$\langle \gamma_t g(x_t), x_t - x_*\rangle \le V_\omega(x_*, x_t) - V_\omega(x_*, x_{t+1}) + \frac{\gamma_t^2}{2}\|g(x_t)\|_*^2 \tag{15.6}$$

By convexity, we have $\gamma_t(f(x_t) - f(x_*)) \le \langle \gamma_t g(x_t), x_t - x_*\rangle \le V_\omega(x_*, x_t) - V_\omega(x_*, x_{t+1}) + \frac{\gamma_t^2}{2}\|g(x_t)\|_*^2$. Taking summation leads to

$$\sum_{t=1}^{T} \gamma_t(f(x_t) - f(x_*)) \le V_\omega(x_*, x_1) + \frac{1}{2}\sum_{t=1}^{T}\gamma_t^2\|g(x_t)\|_*^2$$

Notice that

$$\sum_{t=1}^{T} \gamma_t(f(x_t) - f(x_*)) \ge \sum_{t=1}^{T}\gamma_t \min_{t=1,\dots,T}(f(x_t) - f(x_*))$$

and that

$$\sum_{t=1}^{T} \gamma_t(f(x_t) - f(x_*)) \ge \sum_{t=1}^{T}\gamma_t \cdot [f(\hat{x}_T) - f(x_*)]$$

where $\hat{x}_T = \frac{\sum_{t=1}^T \gamma_t x_t}{\sum_{t=1}^T \gamma_t}$. we can see (15.4) and (15.5) hold ■

15.3.3 Mirror Decent vs. Subgradient Decent

Let f be convex, with proper choice of stepsize as before, we can see that the convergence of these two methods looks very similar.

- For Subgradient Decent, $\epsilon_t \sim O(\frac{\sqrt{\Omega_s} M_s}{\sqrt{t}})$,
  where $\Omega_s = \max_{x\in X} \frac{1}{2}\|x - x_1\|_2^2$ and $M_s = M_{\|\cdot\|_2}(f) = \max_{x\in X}\|g(x)\|_2$.
- For Mirror Decent, $\epsilon_t \sim O(\frac{\sqrt{\Omega_m} M_m}{\sqrt{t}})$,
  where $\Omega_m = \max_{x\in X} V_\omega(x, x_1)$ and $M_m = M_{\|\cdot\|}(f) = \max_{x\in X}\|g(x)\|_*$.

The rate remains the same, but the constants corresponding to Ω and M differ. In some cases, one can show that using Mirror Descent significantly improves upon the constant.

Case 1. Consider $X = \{x \in \mathbb{R}^n : x_i \ge 0, \sum_{i=1}^n x_i = 1\}$

- For subgradient decent, under ℓ_2 setup, $\omega(x) = \frac{1}{2}\|x\|_2^2$, $\|x\| = \|x\|_* = \|x\|_2$, we know $\Omega_s = 1$.
- For mirror decent, under ℓ_1 setup, $w(x) = \sum_{i=1}^n x_i \ln x_i$, $\|x\| = \|x\|_1, \|x\|_* = \|x\|_\infty$, if we choose $x_1 = \operatorname{argmin}_{x\in X}\omega(x)$, then $\Omega_m \le \max_{x\in X}\omega(x) - \min_{x\in X}\omega(x) = 0 - (-\ln n) = \ln(n)$.

Therefore, the ratio between the efficiency estimates of GD and MD is

$$O\left(\frac{1}{\ln(n)} \cdot \frac{\max_{x\in X}\|g(x)\|_2}{\max_{x\in X}\|g(x)\|_\infty}\right)$$

Note that $\|g(x)\|_\infty \leq \|g(x)\|_2 \leq \sqrt{n}\|g(x)\|_\infty$. Hence, in the worst case, we can see mirror decent would be $O(\sqrt{n})$ faster than subgradient descent.

Case 2. Consider $X = \{x \in \mathbb{R}^n : x_i \geq 0, \|x\|_2 \leq 1\}$. In this case, for sub-gradient method, $\Omega_s = 1$; for mirror decent, $\Omega_m \leq O(\sqrt{n}\ln(n))$. Therefore, Ω_m is $\sqrt{n}$ larger than Ω_s, this offset the effect between M_m and M_s. In this case, mirror decent is not necessarily faster than subgradient decent.

References

[1] Anatoli Juditsky and Arkadi Nemirovski, "First Order Methods for Nonsmooth Convex Large-Scale Optimization, I: General Purpose Methods."

[2] A.S. Nemirovsky and D.B. Yudin, "Problem Complexity and Method Efficiency in Optimization," *SIAM Review Volume 27 Issue 2*, 1985, pp. 264.

IE 598: Big Data Optimization **Fall 2016**

Lecture 16: Smoothing Techniques I – October 18

Lecturer: Niao He *Scribers: Harsh Gupta*

Overview. We discussed Subgradient Descent and Mirror Descent algorithms for non-smooth convex optimization in the past week. We observed that Subgradient Descent is a special case of the Mirror Descent algorithm. But, both these algorithms have general formulations and don't exploit the structure of the problem at hand. In practice, we always know some thing about the structure of the optimization problem we intend to solve. One can then utilize this structure to come up with more efficient algorithms as compared to Subgradient Descent and Mirror Descent algorithms.

16.1 Introduction

We intend to solve the following optimization problem:

$$\min_{x \in X} f(x) \tag{16.1}$$

where f is a convex but non-smooth, i.e., non-differentiable function, and X is a convex compact set. One intuitive way to approach the above problem is to approximate the non-smooth function $f(x)$ by a smooth and convex function $f_\mu(x)$, so that we can use the standard techniques learnt so far in the course to solve the problem. Hence, we want to reduce the problem in (16.1) to the following:

$$\min_{x \in X} f_\mu(x) \tag{16.2}$$

where $f_\mu(x)$ is a L_μ-Lipschitz continuous, smooth and convex approximation of the function $f(x)$. Now we can use the techniques learnt earlier in this course like gradient descent, accelerated gradient descent, Frank Wolfe algorithm, coordinate descent etc., to solve the above problem. Clearly, the objective now is to come up with a reasonably good approximation f_μ of f so that solving (16.2) is as close to solving (16.1) as possible.

A motivation example: Consider the simplest non-smooth and convex function, $f(x) = |x|$. The following function, known as the *Huber function*

$$f_\mu(x) = \begin{cases} \frac{x^2}{2\mu}, |x| \leq \mu \\ |x| - \frac{\mu}{2}, |x| > \mu \end{cases} \tag{16.3}$$

is a smooth approximation of the absolute value function. We plot the two functions (for $\mu = 1$) in Figure 1. We make the following observations:

1. $f_\mu(x)$ is clearly continuous and differentiable everywhere. This can be seen straightforwardly from its formulation given in (16.3).
2. We observe that $f_\mu(x) \leq f(x)$. Also, $f_\mu(x) \geq |x| - \frac{\mu}{2}$, therefore:
$$f(x) - \frac{\mu}{2} \leq f_\mu(x) \leq f(x)$$
Hence, if $\mu \to 0$, then $f_\mu(x) \to f(x)$. Therefore, μ characterizes the approximation accuracy.
3. We also observe that $|f''_\mu(x)| \leq \frac{1}{\mu}$. This implies that $f_\mu(x)$ is $\frac{1}{\mu}$-Lipschitz continuous.

The Hubert function approximation has been widely used in machine learning to approximate non-smooth loss functions, e.g. absolute loss (robust regression), hinge loss (SVM), etc.

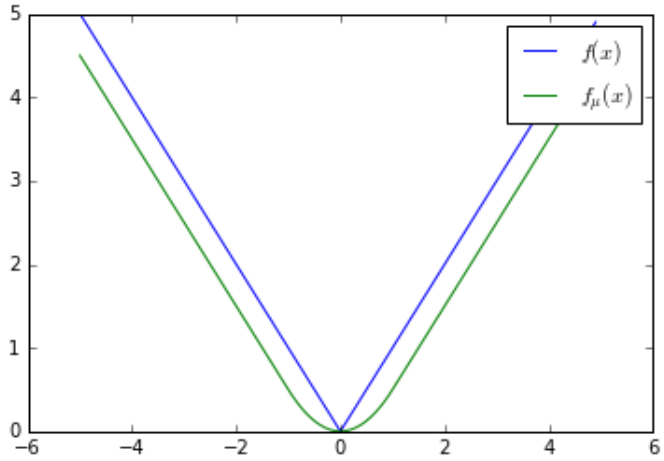


Figure 16.1: Plot for Example 1 ($\mu = 1$)

Robust Regression. Suppose we have m data samples $(a_1, b_1), ..., (a_m, b_m)$. We intend to solve the following regression problem with absolute loss:

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} |{a_i}^T x - b_i|$$

We can approximate the absolute loss in the above optimization problem with the Huber loss and solve instead the following smooth convex optimization problem:

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{m} f_\mu({a_i}^T x - b_i).$$

16.2 Smoothing Techniques

In this section, we will briefly introduce the major smoothing techniques used for non-smooth convex optimization.

1. Nesterov's Smoothing based on conjugate [Nesterov, 2005]

Nesterov's smoothing technique uses the following function to approximate $f(x)$:

$$f_\mu(x) = \max_{y \in \text{dom}(f^*)} \{x^T y - f^*(y) - \mu \cdot d(y)\} \tag{16.4}$$

where f^* is the convex conjugate of f defined as the following:

$$f^*(y) = \max_{x \in \text{dom}(f)} \{x^T y - f(x)\} \tag{16.5}$$

and $d(y)$ is some proximity function that is strongly convex and nonnegative everywhere.

2. Moreau-Yosida smoothing/regularization

Moreau-Yosida's smoothing technique uses the following function to approximate $f(x)$:

$$f_\mu(x) = \min_{y \in \text{dom}(f)} \{f(y) + \frac{1}{2\mu} ||x - y||_M^2\} \tag{16.6}$$

where $\mu > 0$ is the approximation parameter, and the M-norm is defined as $||x||_M^2 = x^T M x$.

3. Ben-Tal-Teboulle smoothing based on recession function [Ben-Tal and Teboulle, 1989]

Ben-Tal and Teboulli's smoothing technique is applicable on a particular class of function which can be represented as following:

$$f(x) = F(f_1(x), f_2(x), ..., f_m(x)) \tag{16.7}$$

where $F(y) = \max_{x \in \text{dom}(g)}\{g(x+y) - g(x)\}$ is the *recession function* of some function $g : \mathbb{R}^m \to \mathbb{R}$ here. For a function f satisfying the above condition, the Ben-Tal and Teboulli's smoothing technique uses the following function to approximate $f(x)$:

$$f_\mu(x) = \mu g\left(\frac{f_1(x)}{\mu}, ..., \frac{f_m(x)}{\mu}\right) \tag{16.8}$$

4. Randomized smoothing [Duchi, Bertlett, and Wainwright, 2012]

The randomized smoothing paradigm uses the following function to approximate $f(x)$:

$$f_\mu(x) = \mathbb{E}_Z f(x + \mu Z) \tag{16.9}$$

where Z is an isotopic Gaussian or uniform random variable.

In this lecture, we will mainly discuss Nesterov's smoothing and we will first discuss some conjugate theory in order to gain insight into this smoothing technique.

16.3 Conjugate Theory

Definition (Convex Conjugate). For any function $f : \mathbb{R}^n \to \mathbb{R}$, its convex conjugate is given as:

$$f^*(y) = \sup_{x \in \text{dom}(f)} \{x^T y - f(x)\} \tag{16.10}$$

Note that f need not necessarily be convex for the above definition. Also, note that f^* will always be convex (regardless of f) since it is the supremum over linear functions of y.

By definition, we have:

$$f^*(y) \geq x^T y - f(x), \forall x, y \quad \Rightarrow \quad x^T y \leq f(x) + f^*(y), \forall x, y$$

The last inequality above is known as the Fenchel inequality. In the previous lectures, we studied the Young's inequality:

$$x^T y \leq \frac{||x||^2}{2} + \frac{||y||_*^2}{2}, \forall x, y$$

Note that the Young's inequality is essentially a special case of the Fenchel inequality. We now present the Moreau-Fenchel duality in the form of the following lemma.

Lemma 16.1 *If f is convex, lower semi-continuous and proper, then $(f^*)^* = f$.*

Here lower semi-continuity means that $\lim_{x \to x_0} \inf f(x) \geq f(x_0)$. In other words, the level set $(\{x : f(x) \leq \alpha\})$ of f is a closed set. A proper convex function means that $f(x) > -\infty$.

Thus for f satisfying the lemma, $f(x)$ admits the Fenchel representation

$$f(x) = \max_{y \in \text{dom} f^*} \{y^T x - f^*(y)\}.$$

Proposition 16.2 *If f is μ-strongly convex then f^* is continuously differentiable and $\frac{1}{\mu}$-Lipschitz smooth.*

Proof:

By definition, we have $f^*(y) = \sup_{x\in \text{dom}(f)}\{y^T x - f(x)\}$. This give us the subdifferential set

$$\partial f^*(y) = \arg\max_{x\in \text{dom} f}\{y^T x - f(x)\}$$

Note that for all y, the optimal solution of the above problem is unique due to strong convexity. Hence, $\partial f^*(y)$ is a single set, i.e. $\partial f^*(y) = \nabla f^*(x)$. Hence, f^* is continuously differentiable. Now, we need to show the following:

$$||\bigtriangledown f^*(y_1) - \bigtriangledown f^*(y_2)||_2 \leq \frac{1}{\mu}||y_1 - y_2||_2, \forall y_1, y_2 \tag{16.11}$$

Let $x_1 = \arg\max_{x\in \text{dom} f}\{y_1^T x - f(x)\}$. Similarly, let $x_2 = \arg\max_{x\in \text{dom} f}\{y_2^T x - f(x)\}$. From the optimality condition, we get:

$$\langle y_1, x_2 - x_1\rangle \leq \langle \partial f(x_1), x_2 - x_1\rangle \tag{16.12}$$

$$\langle y_2, x_1 - x_2\rangle \leq \langle \partial f(x_2), x_1 - x_2\rangle \tag{16.13}$$

From the μ-strong convexity of f, we have:

$$f(x_2) \geq f(x_1) + \partial f(x_1)^T(x_2 - x_1) + \frac{\mu}{2}||x_2 - x_1||_2^2 \tag{16.14}$$

$$f(x_1) \geq f(x_2) + \partial f(x_2)^T(x_1 - x_2) + \frac{\mu}{2}||x_1 - x_2||_2^2 \tag{16.15}$$

Combining equations (16.12), (16.13) with (16.14), (16.15) we get:

$$\langle y_1 - y_2, x_1 - x_2\rangle \geq \mu||x_1 - x_2||_2^2.$$

From the Cauchy-Schwarz inequality, this further implies that

$$\Rightarrow ||x_1 - x_2|| \leq \frac{1}{\mu}||y_1 - y_2||$$

Hence, (16.11) follows from the definitions of x_1, x_2. ■

Remark. Recall the Nesterov's smoothing technique

$$f_\mu(x) = \max_{y\in \text{dom}(f^*)}\{x^T y - f^*(y) - \mu \cdot d(y)\} = (f^* + \mu d)^*(x)$$

by adding the strongly convex term $\mu d(y)$ term, the function $(f^* + \mu d)$ is strongly convex. Therefore, function $f_\mu(x)$ continuously differentiable and Lipschitz-smooth.

16.4 Nesterov's Smoothing

We consider a more generalized problem setting as compared to the previous sections. The goal is to solve the nonsmooth convex optimization problem

$$\min_{x\in X} f(x)$$

We assume that function f can be represented by

$$f(x) = \max_{y\in Y}\{\langle Ax + b, y\rangle - \phi(y)\}$$

where $\phi(y)$ is a convex and continuous function and Y is a convex and compact set. Note that the aforementioned representation generalizes the Fenchel representation using conjugate function and needs not be unique. Indeed, for many cases, we are able to construct such representation easily as compared to using the convex conjugate.

Example. Let $f(x) = \max_{1\le i\le m} |a_i^T x - b_i|$. Computing the convex conjugate for f is a cumbersome task and f^* turns out to be very complex. But we can easily represent f as follows:

$$f(x) = \max_{y\in\mathbb{R}^m}\{(a_i^T x - b_i)y_i : \sum_i |y_i| \le 1\}.$$

We now proceed to discuss some properties of the proximity function (or *prox function*) $d(y)$.

Proximity Function The function $d(y)$ should satisfy the following properties:

- $d(y)$ is continuous and 1-strongly convex on Y;
- $d(y_0) = 0$, for $y_0 \in \arg\min_{y\in Y} d(y)$;
- $d(y) \ge 0, \forall y \in Y$.

Let $y_0 \in Y$, here are some examples of valid prox functions:

- $d(y) = \frac{1}{2}||y - y_0||_2^2$
- $d(y) = \frac{1}{2}\sum w_i(y_i - (y_0)_i)^2$ with $w_i \ge 1$
- $d(y) = \omega(y) - \omega(y_0) - \nabla\omega(y_0)^T(y - y_0)$ with $\omega(x)$ being 1-strongly convex on Y

We can check that these proximity function also satisfies all the properties mentioned above.

Nesterov's smoothing considers the following smooth approximation of f

$$f_\mu(x) = \max_{y\in Y}\{\langle Ax + b, y\rangle - \phi(y) - \mu\cdot d(y)\}.$$

We first describe below the Lipschitz smoothness of this function [Nesterov, 2005].

Proposition 16.3 *For $f_\mu(x)$, we have*

- *$f_\mu(x)$ is continuously differentiable.*
- *$\bigtriangledown f_\mu(x) = A^T y(x)$, where $y(x) = \arg\max_{y\in Y}\{\langle Ax + b, y\rangle - \phi(y) - \mu\cdot d(y)\}$.*
- *$f_\mu(x)$ is $\frac{||A||_2^2}{\mu}$ Lipschitz smooth, where $\|A\|_2 := \max_x\{\|Ax\|_2 : \|x\|_2 \le 1\}$.*

This can be derived similarly as Proposition 16.2, we omit the proofs here.

Now let us look at the approximation accuracy.

Theorem 16.4 *For any $\mu > 0$, let $D_Y^2 = \max_{y\in Y} d(y)$, we have*

$$f(x) - \mu D_Y^2 \le f_\mu(x) \le f(x).$$

This follows directly from the fact that $0 \le d(y) \le D_Y^2$.

Remark. Let $f_* = \min_{x\in X} f(x)$ and $f_{\mu,*} = \min_{x\in X} f_\mu(x)$, we have $f_{\mu,*} \le f_*$. Moreover, for any x_t generated by an algorithm

$$f(x_t) - f_* \le \underbrace{f(x_t) - f_\mu(x_t)}_{\text{approximation error}} + \underbrace{f_\mu(x_t) - f_{\mu,*}}_{\text{optimization error}}$$

Suppose we have access to compute the gradient of $f_\mu(x)$ when solving the resulting smooth convex optimization problem

$$\min_{x\in X} f_\mu(x)$$

(i) If we apply projected gradient descent to solve the smooth problem, then we have:

$$f(x_t) - f_* \le O(\frac{||A||_2^2 D_X^2}{\mu t} + \mu D_Y^2)$$

Therefore, if we want the error to be less than a threshold ϵ, we need to set $\mu = O(\frac{\epsilon}{D_Y^2})$ and the total number of iterations is at most $T_\epsilon = O(\frac{||A||_2^2 D_X^2}{\epsilon\mu}) = O(\frac{||A||_2^2 D_X^2 D_Y^2}{\epsilon^2})$.

(ii) If we apply accelerated gradient descent to solve the smooth problem, then we have

$$f(x_t) - f_* \le O(\frac{||A||_2^2 D_X^2}{\mu t^2} + \mu D_Y^2)$$

Therefore, if we want the error to be less than a threshold ϵ, we need to set $\mu = O(\frac{\epsilon}{D_Y^2})$ and the total number of iterations is at most $T_\epsilon = O(\frac{||A||_2 D_X}{\sqrt{\epsilon\mu}}) = O(\frac{||A||_2 D_X D_Y}{\epsilon})$.

In the latter case, the overall complexity $O(1/\epsilon)$ is substantially better than the $O(1/\epsilon^2)$ complexity when we directly apply subgradient descent to solve the original nonsmooth convex problem.

References

[Nes05] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*,Ser. A 103(1), 127–152 (2005)

[Ben89] A. Ben-Tal, and M. Teboulle, "A smoothing technique for nondifferentiable optimization problems." In *Optimization*, pp. 1-11 (1989)

[Lem97] C. Lemaréchal and C. Sagastizábal, " Practical aspects of the Moreau–Yosida regularization: Theoretical preliminaries". SIAM Journal on Optimization, 7(2), 367-385 (1997)

[DBW12] J.C. Duchi, P.L. Bartlett, and M.J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM J. OPTIM.*,Vol. 22, No. 2, pp. 674–701 (2012)

IE 598: Big Data Optimization **Fall 2016**

Lecture 17: Smoothing Techniques II – October 20

Lecturer: Niao He *Scriber: Sai Kiran Burle*

Overview. In the last lecture, we discussed several smoothing techniques to approximate nonsmooth convex functions, particularly Nesterov's smoothing technique. In this lecture, we will drill down the algorithmic details of gradient descent when applied to solve the smooth counterpart and discuss its connection to proximal point algorithm.

17.0 Recap of Nesterov's Smoothing

Aside from Subgradient Descent or Mirror Descent, using smoothing technique is another way to address nonsmooth convex optimization problems:

$$\min_{x\in X} f(x)$$

Suppose function f can be represented as

$$f(x) = \sup_{y\in Y} \{\langle Ax+b, y\rangle - \phi(y)\}$$

where Y is a convex, compact set and ϕ is a convex function. Nesterov's smoothing of f is given by

$$f_\mu(x) = \sup_{y\in Y} \{\langle Ax+b, y\rangle - \phi(y) - \mu d(y)\} = (\phi + \mu d)^* (Ax+b)$$

where $\mu > 0$ is the smoothing parameter, function $d(.)$ is strongly convex, everywhere nonnegative and $\min_{y\in Y} d(y) = 0$.

From the last lecture, we show that when applying the accelerated gradient descent to solve the smoothing counterpart:

$$\min_{x\in X} f_\mu(x)$$

we arrive at

$$f(x_t) - f_* \le \underbrace{[f(x_t) - f_\mu(x_t)]}_{\text{approximation error}} + \underbrace{[f_\mu(x_t) - f_{\mu,x}]}_{\text{optimization error}} \le \mathcal{O}\left(\mu D_Y^2 + \frac{\|A\|_2^2 D_X^2}{\mu t^2}\right)$$

In order to achieve ϵ accuracy, one should choose $\mu = \mathcal{O}\left(\frac{\epsilon}{D_Y^2}\right)$, and this leads to the an overall iteration complexity $\mathcal{O}\left(\frac{\|A\|_2 D_X D_Y}{\epsilon}\right)$. Note that this is substantially better than the $O(1/\epsilon^2)$ complexity required by Subgradient Descent or Mirror Descent.

17.1 Some Examples

We mentioned that Nesterov's smoothing technique is more general and flexible comparing to other smoothing techniques, such as Moreau-Yosida regularization, Teboulle's smoothing based on recession function. We

provide below a simple example to illustrate the smoothed function under difference choices of proximity function $d(\cdot)$ and Fenchel representation.

Example: $f(x) = |x|$
Note that f admits the following two different representation:

$$f(x) = \sup_{|y|\le 1} yx$$
$$\text{OR} \qquad f(x) = \sup_{\substack{y_1,y_2\ge 0\\ y_1+y_2=1}} (y_1 - y_2)x$$

Here, $Y = \{y : |y| \le 1\}$ or $Y = \{y = (y_1, y_2) : y_1, y_2 \ge 0, y_1 + y_2 = 1\}$, function $\phi(y) \coloneqq 0$.

Now we consider different choices for the distance function $d(y)$.

1. $d(y) = \frac{1}{2}y^2$. Clearly, $d(.)$ is 1-strongly convex on $Y = \{y : |y| \le 1\}$, and $d(y) \ge 0$.
   Nesterov's smoothing gives rise to

$$f_\mu(x) = \sup_{|y|\le 1} \left\{yx - \frac{\mu}{2}y^2\right\} = \begin{cases} \frac{x^2}{2\mu}, & |x| \le \mu \\ |x| - \frac{\mu}{2}, & |x| > \mu \end{cases} \tag{17.1}$$

   which is the well-known Huber function.

   Remark. The same approximation can be obtained from Moreau-Yosida smoothing technique as follows:

$$f_\mu(x) = \inf_{y\in Y} \left\{|y| + \frac{1}{2\mu}\|y - x\|_2^2\right\}$$

2. $d(y) = 1 - \sqrt{1 - y^2}$. Clearly, $d(.)$ is 1-strongly convex on $Y = \{y : |y| \le 1\}$ and $d(y) \ge 0$.
   Nesterov's smoothing gives rise to

$$f_\mu(x) = \sup_{|y|\le 1} \left\{yx - \mu\left(1 - \sqrt{1 - y^2}\right)\right\} = \sqrt{x^2 + \mu^2} - \mu \tag{17.2}$$

   Remark. The same approximation can be obtained from Ben-Ta l&Teboulle's smoothing based on recession function:

$$|x| = \sup_y \{g(x + \mu) - g(y)\}, \quad g(y) = \sqrt{1 + y^2}$$
$$f_\mu(x) = \mu g\left(\frac{x}{\mu}\right) = \sqrt{x^2 + \mu^2}$$

3. $d(y) = y_1 \log y_1 + y_2 \log y_2 + \log 2$. Clearly, $d(.)$ is 1-strongly convex on $Y = \{(y_1, y_2) : y_1, y_2 \ge 0, y_1 + y_2 = 1\}$ and $d(y) \ge 0$.
   Nesterov's smoothing gives rise to

$$f_\mu(x) = \sup_{\substack{y_1,y_2\ge 0\\ y_1+y_2=1}} \{(y_1 - y_2)x - \mu(y_1 \log y_1 + y_2 \log y_2 + \log 2)\} = \mu \log\left(\frac{e^{-\frac{x}{\mu}} + e^{\frac{x}{\mu}}}{2}\right) \tag{17.3}$$

Remark. The same approximation can be obtained from Ben-Tal Teboulle smoothing based on recession function.

$$|x| = \max\{x, -x\} = \sup_y \{g(x+\mu) - g(y)\}, \quad g(y) = \log\left(e^{y_1} + e^{y_2}\right)$$
$$f_\mu(x) = \mu g\left(\frac{x}{\mu}\right) = \mu \log\left(e^{-\frac{x}{\mu}} + e^{\frac{x}{\mu}}\right)$$

17.2 Nesterov's smoothing and Moreau-Yosida regularization

In the previous example, we see that under the simple proximity function $d(y) = \frac{1}{2}\|y\|_2^2$, Nesterov's smoothing is equivalent to Yoreau-Yosida regularization. Indeed, this is true in general. Let f^* denote the conjugate of the function f. Suppose f is proper, convex and lower-semicontinuous, then

$$f(x) = \max_y \left\{y^T x - f^*(y)\right\}$$

Then we can show that

$$\begin{array}{rcll} f_\mu(x) & = & \max_y \left\{y^T x - f^*(y) - \frac{\mu}{2}\|y\|_2^2\right\} & (\textbf{Nesterov's smoothing}) \\ & = & \left(f^* + \frac{\mu}{2}\|\cdot\|_2^2\right)^*(x) & \\ & = & \inf_y \left\{f(y) + \frac{1}{2\mu}\|x-y\|_2^2\right\} & (\textbf{ Moreau-Yousida regularization}) \end{array} \tag{17.4}$$

where the last equation follows from the following Lemma 17.1(a).

Lemma 17.1 *Let f and g be two proper, convex and semi-continuous functions, then*

(a) $(f+g)^*(x) = \inf_y \{f^*(y) + g^*(x-y)\}$

(b) $(\alpha f)^*(x) = \alpha f^*\left(\frac{x}{\alpha}\right)$ *for* $\alpha > 0$

Proof:

(a) First, we prove the following equality

$$(f \square g)^*(x) = f^*(x) + g^*(x)$$

where $f \square g$ denotes the convlution operator,

$$f \square g = \inf_y \{f(y) + g(x-y)\}$$

$$\begin{aligned} (f\square g)^*(x) &= \sup_z \left\{z^T x - \inf_y \left(f(y) + g(z-y)\right)\right\} \\ &= \sup_z \left\{z^T x - \inf_{y_1+y_2=z} \left(f(y_1) + g(y_2)\right)\right\} \\ &= \sup_z \left\{\sup_{y_1+y_2=z} \left\{(y_1+y_2)^T x - f(y_1) - g(y_2)\right\}\right\} \\ &= \sup_{y_1,y_2} \left\{(y_1+y_2)^T x - f(y_1) - g(y_2)\right\} \\ &= \sup_{y_1} \left\{y_1^T x - f(y_1)\right\} + \sup_{y_2} \left\{{y_2}^T x - g(y_2)\right\} \\ &= f^*(x) + g^*(x) \end{aligned}$$

So, we get

$$(F\square G)^*(x) = F^*(x) + G^*(x)$$

Using $F = f^*$, and $G = g^*$,

$$(f^*\square g^*)^*(x) = f(x) + g(x)$$

Taking conjugate on both sides, we arrive at

$$(f^*\square g^*)(x) = (f+g)^*(x)$$

This leads to the desired result.

(b) This is because

$$\begin{aligned}(\alpha f)^*(x) &= \sup_y \left\{y^T x - \alpha f(y)\right\} \\ &= \alpha \sup_y \left\{y^T \left(\frac{x}{\alpha}\right) - f(y)\right\} \\ &= \alpha f^*\left(\frac{x}{\alpha}\right)\end{aligned}$$

■

17.3 Proximal Point Algorithms

Note that when computing the gradient of the smoothed function $\nabla f_\mu(x)$, for any smoothing forms used in (17.4), we will need to solve subproblems in the form

$$\min_y \left\{f(y) + \frac{1}{2}\|x-y\|_2^2\right\}.$$

The optimal solution to this subproblem is often referred to as the proximal operator, which shares many similarity as the projection operator we discussed earlier. We provide some basic results below.

17.3.1 Basics of Proximal Operators

Definition 17.2 *Given a convex function f, the* **proximal operator** *of f at a given point x is defined as*

$$prox_f(x) = \arg\min_y \left\{f(y) + \frac{1}{2}\|x-y\|_2^2\right\}$$

As an immediate observation, for any $\mu > 0$, we have

$$prox_{\mu f}(x) = \arg\min_y \left\{f(y) + \frac{1}{2\mu}\|x-y\|_2^2\right\}$$

Example: Let f be the indicator function of a convex set X, namely,

$$f(x) = \delta_X(x) = \begin{cases}0, & x \in X \\ +\infty, & x \notin X\end{cases}$$

Then the proximal operator reduces to the projection operator onto X, i.e.

$$prox_f(x) = \arg\min_{y \in X} \left\{ \frac{1}{2} \|x - y\|_2^2 \right\} = \Pi_X(x).$$

In general, the proximal operator possesses many similar properties as the projection operator as discussed earlier, e.g. treating optimal solution as fixed point, non-expansiveness, and decomposition.

Proposition 17.3 *Let f be a convex function, we have*

(a) **(Fixed Point)** *A point x_* minimizes $f(x)$ iff $x_* = prox_f(x_*)$.*

(b) **(Non-expansive)** $\|prox_f(x) - prox_f(y)\|_2 \leq \|x - y\|_2$.

(c) **(Moreau Decomposition)** *For any x, $x = prox_f(x) + prox_{f^*}(x)$.*

Proof:

(a) First, if x_* minimizes $f(x)$, we have $f(x) \geq f(x_*), \forall x \in X$. Hence,

$$f(x) + \frac{1}{2} \|x - x_*\|_2^2 \geq f(x_*) + \frac{1}{2} \|x_* - x_*\|_2^2$$

This implies that

$$x_* = \arg\min_x \left\{ f(x) + \frac{1}{2} \|x - x_*\|_2^2 \right\} = prox_f(x_*)$$

To prove the converse, consider if

$$x_* = prox_f(x_*) = \arg\min_x \left\{ f(x) + \frac{1}{2} \|x - x_*\|_2^2 \right\}$$

By the optimality condition, this implies that

$$0 \in \partial f(x_*) + (x_* - x_*) \implies 0 \in \partial f(x_*)$$

Therefore, x_* minimizes f.

(b) Let us denote $u_x = \text{prox}_f(x)$ and $u_y = \text{prox}_f(x_y)$. Equivalently,

$$x - u_x \in \partial f(u_x) \quad \text{and} \quad y - u_y \in \partial f(u_y).$$

Now we use the fact that ∂f is a monotone mapping, which tells us that

$$\langle x - u_x - (y - u_y), u_x - u_y \rangle \geq 0$$

Hence, we have

$$\langle x - y, u_x - u_y \rangle \geq \|u_x - u_y\|_2^2 .$$

By Cauchy Schwartz inequality, this leads to $\|u_x - u_y\|_2 \leq \|x - y\|_2$ as desired.

(c) Let $u = prox_f(x)$, or equivalently, $x - u \in \partial f(u)$. Note that we also have $u \in \partial f^*(x - u)$, this is equivalent to $x - u = prox_{f^*}(x)$. Hence, $x = u + (x - u) = prox_f(x) + prox_{f^*}(x)$.

■

17.3.2 Gradient Descent for Smoothed Function

Recall the definition of $f_\mu(x)$ in (17.4), we can see that the gradient is given by

$$\nabla f_\mu(x) = \frac{1}{\mu}(x - prox_{\mu f}(x)) \tag{17.5}$$

Since f_μ is $\frac{1}{\mu}$-smooth, gradient descent works as follows

$$x_{t+1} = x_t - \mu \nabla f_\mu(x_t)$$

From equation (17.5), this is equivalent as

$$x_{t+1} = prox_{\mu f}(x_t)$$

which is known as **proximal point algorithm**, initially proposed by Rockafellar in 1976. We discuss below the general algorithm and its convergence results.

17.3.3 Proximal Point Algorithm [RT76]

The goal is to minimize a non-smooth convex function $f(x)$, i.e. $\min_x f(x)$. The proximal point algorithm works as follows:

$$x_{t+1} = prox_{\gamma_t f}(x_t) \qquad t = 0, 1, 2, ...$$

where $\gamma_t > 0$ are the stepsizes.

Theorem 17.4 *Let f be a convex function, the proximal point algorithm satisfies*

$$f(x_t) - f_* \leq \frac{\|x_0 - x_*\|_2^2}{2\sum_{\tau=0}^{t-1} \gamma_\tau}$$

Proof: First, by optimality of x_{t+1}:

$$f(x_{t+1}) + \frac{1}{2\gamma_t}\|x_{t+1} - x_t\|_2^2 \leq f(x_t)$$

i.e.,

$$f(x_t) - f(x_{t+1}) \geq \frac{1}{2\gamma_t}\|x_{t+1} - x_t\|_2^2.$$

This further implies that $f(x_t)$ is non-increasing at each iteration. Note that this is essentially the key inequality when we analyze the convergence for Gradient Descent in Lecture 8. By far, we can simply adopt that analysis to get the desired result. To make it self-contained, we provide the full proof here.

Let $g \in \partial f(x_{t+1})$, by convexity of f, we have $f(x_{t+1}) - f_* \leq g^T(x_{t+1} - x_*)$. From the optimality condition of x_{t+1}, we have

$$0 \in \partial f(x_{t+1}) + \frac{1}{\gamma_t}(x_{t+1} - x_t) \implies \frac{x_t - x_{t+1}}{\gamma_t} \in \partial f(x_{t+1})$$

Hence,

$$\begin{aligned} f(x_{\tau+1}) - f_* &\leq \frac{1}{\gamma_\tau}(x_\tau - x_{\tau+1})^T(x_{\tau+1} - x_*) \\ &\leq \frac{1}{\gamma_t}(x_\tau - x_* + x_* - x_{\tau+1})^T(x_{\tau+1} - x_*) \\ &\leq \frac{1}{\gamma_t}\left[(x_\tau - x_*)^T(x_{\tau+1} - x_*) - \|x_{\tau+1} - x_*\|_2^2\right] \end{aligned}$$

Since $(x_\tau - x_*)^T(x_{\tau+1} - x_*) \leq \frac{1}{2}[\|x_\tau - x_*\|_2^2 + \|x_{\tau+1} - x_*\|_2^2]$, this implies that

$$\gamma_\tau(f(x_{\tau+1}) - f_*) \leq \frac{1}{2}\left[\|x_\tau - x_*\|_2^2 - \|x_{\tau+1} - x_*\|_2^2\right]$$

Summing this inequality for $\tau = 0, 1, 2, ...t-1$,

$$\sum_{\tau=0}^{t-1} \gamma_\tau(f(x_{\tau+1}) - f_*) \leq \frac{\|x_0 - x_*\|_2^2}{2} - \frac{\|x_t - x_*\|_2^2}{2} \leq \frac{\|x_0 - x_*\|_2^2}{2}$$

Since $f(x_\tau)$ is non-increasing, we have

$$\left(\sum_{\tau=0}^{t-1} \gamma_\tau\right)(f(x_t) - f_*) \leq \sum_{\tau=0}^{t-1} \gamma_\tau(f(x_{\tau+1}) - f_*)$$

Therefore,

$$f(x_t) - f_* \leq \frac{\|x_0 - x_*\|_2^2}{2\sum_{\tau=0}^{t-1} \gamma_\tau}$$

■

Remark. Note that

1. Unlike most algorithms we discussed so far in this course, the algorithm is not a gradient-based algorithm.
2. γ_t can be arbitrarily, the algorithm converges as long as $\sum_t \gamma_t \to \infty$, however, the cost of the proximal operator will depend on γ_t. For larger γ_t, the algorithm converges faster, but the proximal operator $prox_{\gamma_t f}(x_t)$ might be harder to solve.
3. If $\gamma_t = \mu$ (const.), then $f(x_t) - f_* = \mathcal{O}\left(\frac{1}{\mu t}\right)$. This matches with the $O(1/t)$ rate we obtain from the gradient descent perspective.

17.3.4 Accelerated Proximal Point Algorithm [GO92]

When combined with Nesterov's acceleration scheme, we get the **accelerated proximal point algorithm**, which works as follows

$$\begin{aligned} x_{t+1} &= prox_{\gamma_t f}(y_t) \\ y_{t+1} &= x_{t+1} + \beta_t(x_{t+1} - x_t) \end{aligned}$$

where β_t satisfies that

$$\beta_t = \frac{\alpha_t(1-\alpha_t)}{\alpha_t^2 + \alpha_{t+1}}$$
$$\alpha_{t+1}^2 = (\alpha_{t+1}+1)\alpha_t^2 - \frac{\mu}{L}\alpha_{t+1}$$

With the accelerated proximal point algorithm, the convergence can be improved to

$$f(x_t) - f_* \leq \mathcal{O}\left(\frac{1}{\left(\sum_{\tau=0}^{t-1}\sqrt{\gamma_t}\right)^2}\right)$$

When $\gamma_t = \mu$ (const.), then $f(x_t) - f_* = \mathcal{O}\left(\frac{1}{\mu t^2}\right)$. Again, this matches with the $O(1/t^2)$ rate we obtain from the gradient descent perspective.

References

[GO92] Güler, Osman. "New proximal point algorithms for convex minimization." *SIAM Journal on Optimization* 2.4 (1992): 649-664.

[RT76] Rockafellar, R. Tyrrell. "Monotone operators and the proximal point algorithm." *SIAM journal on control and optimization* 14.5 (1976): 877-898.

IE 598: Big Data Optimization **Fall 2016**

Lecture 18: Mirror-Prox Algorithm – October 25

Lecturer: Niao He *Scriber: Meghana Bande*

Overview: In this lecture, the mirror-prox algorithm is introduced to solve non-smooth convex functions. This involves conversion of the minimization problem to a convex concave saddle point problem. The convergence of the algorithm is also discussed.

18.1 Introduction

Nonsmooth convex optimization. Previously, we have looked at optimization problems of the form $\min_{x\in X} f(x)$, where X is a convex compact set and the function f is convex but non-smooth. We have discussed two approaches to solve this problem.

1. Subgradient descent or the more general version, Mirror descent can be used to solve this problem. These approaches give a rate of convergence of $O(\frac{1}{\sqrt{t}})$, which is indeed optimal among all subgradient-based algorithms.
2. Smoothing approach (e.g., Nesterov's smoothing technique) is to exploit the structure of the function f to find a smooth approximation f_μ and use accelerated gradient descent to solve the optimization problem for this smooth function. These approaches give a convergence rate of $O(\frac{1}{t})$.

Drawbacks of smoothing techniques. The drawbacks with the smoothing techniques discussed previously are the following:

1. The performance of the algorithm is very sensitive to the smoothness parameter μ. The optimal choice of $\mu \sim O(\frac{\epsilon}{D_Y^2})$ cannot be calculated since D_Y and ϵ may not be known. Using a large μ may lead to a bad approximation of the original function f while using a smaller μ may result in slower convergence of the algorithm used.
2. The approach uses the gradient of $\nabla f_\mu(x)$ or a prox operator which involves solving the optimization problem $\min_{x\in X}\{f(x) + \frac{1}{2\mu}||x-y||^2\}$. This can be expensive to calculate in many scenarios.

In this lecture, we will discuss the **mirror-prox method** which does not use any smoothing parameter μ.

If f can be represented as $\max_{y\in Y}\{\langle Ax+b, y\rangle - \phi(y)\}$, instead of solving a smooth approximate function, we can directly solve the minimax function i.e.,

$$\min_{x\in X} f(x) \iff \min_{x\in X}\max_{y\in Y}\{\langle Ax+b, y\rangle - \phi(y)\}.$$

Recall that we encountered minimax problems previously when we used Lagrangian dual to solve constrained optimization problems.

$$\min_{x\in X, g(x)\le 0} f(x) \iff \min_{x\in X}\max_{\lambda\ge 0}\{f(x) + \lambda^T g(x)\}.$$

In this Lagrangian setting, we discussed that i) saddle point exists if the slater condition is satisfied, and ii) if saddle point exists, then the corresponding x solves the primal problem.

18.2 Smooth Convex-Concave Saddle Point Problems

Consider the saddle point problem

$$\min_{x\in X}\max_{y\in Y}\phi(x,y), \tag{18.1}$$

under the following assumptions,

1. For each $y\in Y$, the function $\phi(x,y)$ is convex in the variable x and for each $x\in X$, the function $\phi(x,y)$ is concave in the variable y.
2. The sets X,Y are closed, convex sets.
3. $\phi(x,y)$ is a smooth function, i.e., $\nabla\phi(x,y)=[\nabla_x\phi(x,y),\nabla_y\phi(x,y)]$ is Lipschitz continuous on the domain of $X\times Y$.

A feasible point (x_*,y_*) for (18.1) is a saddle point if

$$\phi(x_*,y)\le\phi(x_*,y_*)\le\phi(x,y_*)\quad\forall x\in X,y\in Y.$$

Lemma 18.1 (Sion's minimax theorem, existence of saddle point) *If one of sets X,Y is bounded then the saddle point to* (18.1) *always exists.*

We now consider the primal and dual optimization problems induced by the convex concave saddle point problem (18.1).

$$\mathrm{Opt(P)}=\min_{x\in X}\bar{\phi}(x),\quad \bar{\phi}(x)=\max_{y\in Y}\phi(x,y)\quad (P)$$

$$\mathrm{Opt(D)}=\max_{y\in Y}\underline{\phi}(y),\quad \underline{\phi}(y)=\min_{x\in X}\phi(x,y)\quad (D)$$

If (x_*,y_*) is the saddle point of (18.1), then x_* is the optimal solution to (P) and y_* is the optimal solution to (D), i.e., we have,

$$\bar{\phi}(x_*)=\mathrm{Opt(P)}=\phi(x_*,y_*)=\mathrm{Opt(D)}=\underline{\phi}(y_*).$$

Given a candidate solution $z=(x,y)$, we quantify the inaccuracy or error by $\epsilon_{\mathrm{sad}}(z)$ defined as

$$\epsilon_{\mathrm{sad}}(z)=\bar{\phi}(x)-\underline{\phi}(y).$$

We note that for all $z\in X\times Y$, $\epsilon_{\mathrm{sad}}(z)\ge 0$, and $\epsilon_{\mathrm{sad}}(z)=0$ iff z is the saddle point.

Since $\mathrm{Opt(P)}=\mathrm{Opt(D)}$, $\epsilon_{\mathrm{sad}}(z)$ can be written as

$$\epsilon_{\mathrm{sad}}(z)=\bar{\phi}(x)-\mathrm{Opt(P)}+\mathrm{Opt(D)}-\underline{\phi}(y),$$

and hence we have,

$$\bar{\phi}(x)-\mathrm{Opt(P)}\le\epsilon_{\mathrm{sad}}(z),$$
$$\mathrm{Opt(D)}-\underline{\phi}(y)\le\epsilon_{\mathrm{sad}}(z).$$

18.3 Examples

We present a few examples to illustrate the conversion of a non-smooth minimization to a smooth convex concave saddle point problem. In each of these examples, we assume that the set X is a closed convex set.

1. $f(x) = \max_{1 \le i \le m} f_i(x)$ where each $f_i(x)$ is smooth and convex for all $1 \le i \le m$. Note that $f(x)$ is the maximum of convex functions and is typically non-smooth.

   This can be written as $f(x) = \max_{y \in \Delta_m} \sum_{i=1}^m y_i f_i(x)$, where the simplex $\Delta_m = \{y : y \ge 0, \sum y_i = 1\}$ is a compact convex set and the function $\phi(x, y)$ is given by

   $$\phi(x, y) = \sum_{i=1}^m y_i f_i(x).$$

   Note that $\phi(x, y)$ is a smooth function since each f_i is smooth and it is concave (linear) in y for any $x \in X$ and convex in x for any fixed $y \in \Delta_m$.

2. $f(x) = ||Ax - b||_p$ where $||.||_p$ denotes the p-norm given by $||x||_p = (\sum_{i=1}^n x_i^p)^{\frac{1}{p}}$. The function $f(x)$ is convex but non-smooth because it is not differentiable at zero.

   This can be written as $f(x) = \max_{||y||_q \le 1} \langle Ax - b, y \rangle$. Here $Y = \{y : ||y||_q \le 1\}$ is a unit q-norm ball, where q is such that $\frac{1}{p} + \frac{1}{q} = 1$ and is a compact convex set and the function $\phi(x, y)$ is given by

   $$\phi(x, y) = \langle Ax - b, y \rangle.$$

   Note that $\phi(x, y)$ is a smooth function that is concave (linear) in y for any $x \in X$ and convex (linear) in x for any fixed $y \in Y$.

   If $p = 1$, we have the case of robust regression and $q = \infty$ in this case. If $p = 2$, we have least squares regression and in this case $q = 2$.

3. $f(x) = \sum_{i=1}^m \max(1 - (a_i^T x) b_i), 0)$ is a convex piecewise linear function which is non-smooth. This is the hinge loss function used widely in support vector machines.

   This can be written as $f(x) = \max_{0 \le y_i \le 1} \sum_{i=1}^m y_i (1 - (a_i^T x) b_i)$, where the set $Y = \{y : 0 \le y_i \le 1, 1 \le i \le m\}$ is a compact convex set and the function $\phi(x, y)$ is given by

   $$\phi(x, y) = \sum_{i=1}^m y_i (1 - (a_i^T x) b_i).$$

   Note that $\phi(x, y)$ is a smooth function that is concave (linear) in y for any $x \in X$ and convex (linear) in x for any fixed $y \in Y$.

18.4 Mirror-Prox Algorithm

18.4.0 High-level Idea.

If we have access to the gradient of the function $\phi(x, y)$, we can use gradient descent type algorithms to solve the saddle point problem, just like what we do for convex minimization problem.

Consider the "gradient type" vector field $F(z)$ defined for each $z = (x, y)$ as

$$F(z) = [\nabla_x \phi(x, y), -\nabla_y \phi(x, y)].$$

Note that since $\phi(x, y)$ is convex in x and concave in y, $\nabla_x \phi(x, y), -\nabla_y \phi(x, y)$ are descent directions.

It can be shown that the first order optimality condition for the saddle point problem (18.1) is given by

$$z_* \text{ is optimal } \iff \langle F(z_*), z - z_* \rangle \geq 0, \quad \forall z \in X \times Y.$$

This is similar to the optimality condition for convex minimization problem where F stands for the gradient or subgradient. Intuitively, we could apply mirror descent algorithm to solve (18.1) as if we were solving a convex minimization problem, by replacing the subgradient with the above vector field. That is, at each iteration, we run

$$z_{t+1} = \operatorname*{argmin}_{z \in X \times Y} \{V(z, z_t) + \langle \gamma_t F(z_t), z \rangle\}, \tag{18.2}$$

where $V(z, z_t)$ is some Bregman distance defined on $X \times Y$. Extending the analysis we have earlier on mirror descent, we can show that $\epsilon_{\text{sad}}(z) \leq O(\frac{1}{\sqrt{t}})$, which implies a slow $O(1/t)$ rate of convergence, similar as what we obtain when using mirror descent to solve convex minimization problems. In the following, we show that with a slight modification of Mirror Descent, we can achieve the $O(\frac{1}{t})$ convergence rate, matching the results given by Nesterov's smoothing technique.

18.4.1 Mirror Prox

Setup. Let $\omega(z) : X \times Y \to \mathbb{R}$ be a distance generating function where ω is 1-strongly convex function w.r.t some norm $||.||$ on the underlying space and is continuously differentiable. The Bregman distance induced by $\omega(\cdot)$ is given as

$$V(z, z') = \omega(z) - \omega(z') - \nabla\omega(z')^T(z - z') \geq \frac{1}{2}||z - z'||^2.$$

Recall we also have the Bregman three-point identity which states that for any $x, y, z \in \text{dom}(\omega)$, we have

$$V(x, z) = V_\omega(x, y) + V_\omega(y, z) - \langle \nabla\omega(z) - \nabla\omega(y), x - y \rangle.$$

We assume that the vector field $F(z) = [\nabla_x \phi(x, y), -\nabla_y \phi(x, y)]$ is Lipschitz continuous with respect to the norm $\|\cdot\|$, namely,

$$\|F(z) - F(z')\|_* \leq L\|z - z'\|$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$.

Mirror Prox algorithm:

- Initialization:
$$z_1 = (x_1, y_1) \in X \times Y.$$
- Update at each iteration t:
$$\hat{z}_t = (\hat{x}_t, \hat{y}_t) = \operatorname*{argmin}_{z \in X \times Y} \{V(z, z_t) + \langle \gamma_t F(z_t), z \rangle\}, \tag{18.3}$$
$$z_{t+1} = (x_{t+1}, y_{t+1}) = \operatorname*{argmin}_{z \in X \times Y} \{V(z, z_t) + \langle \gamma_t F(\hat{z}_t), z \rangle\}. \tag{18.4}$$

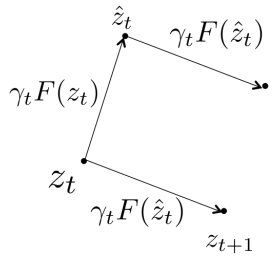


Figure 18.1:

Note that this is not the same as two consecutive steps of the mirror descent algorithm since the first term in the minimization, $V(z, z_t)$ is same in both the steps of the update. This is illustrated in Figure 18.1.

Theorem 18.2 (NEM '04) *Denote the diameter of the Bregman distance* $\Omega = \max_{z \in X \times Y} V(z, z_1)$. *The Mirror Prox algorithm with step-size* $\gamma_t \leq \frac{1}{L}$ *satisfies*

$$\epsilon_{sad}(\bar{z}_T) \leq \frac{\Omega}{\sum_{t=1}^T \gamma_t}, \quad \textit{where } \bar{z}_T = \frac{\sum_{t=1}^T \gamma_t \hat{z}_t}{\sum_{t=1}^T \gamma_t}.$$

Proof: From the Bregman three-point identity and the optimality condition for $\hat{z}_t$ to be the solution of (18.3), we have

$$\langle \gamma_t F(z_t), \hat{z}_t - z \rangle \leq V(z, z_t) - V(z, \hat{z}_t) - V(\hat{z}_t, z_t), \quad \forall z \in X \times Y. \tag{18.5}$$

Similarly optimality at z_{t+1} for (18.4) gives

$$\langle \gamma_t F(\hat{z}_t), z_{t+1} - z \rangle \leq V(z, z_t) - V(z, z_{t+1}) - V(z_{t+1}, z_t), \quad \forall z \in X \times Y. \tag{18.6}$$

Set $z = z_{t+1}$ in (18.5) to obtain

$$\langle \gamma_t F(z_t), \hat{z}_t - z_{t+1} \rangle \leq V(z_{t+1}, z_t) - V(z_{t+1}, \hat{z}_t) - V(\hat{z}_t, z_t). \tag{18.7}$$

Combing (18.6) and (18.7), we have

$$\begin{aligned} \langle \gamma_t F(\hat{z}_t), \hat{z}_t - z \rangle &= \langle \gamma_t F(\hat{z}_t), \hat{z}_t - z_{t+1} \rangle\} + \langle \gamma_t F(\hat{z}_t), z_{t+1} - z \rangle \\ &= \gamma_t \langle F(\hat{z}_t) - F(z_t), \hat{z}_t - z_{t+1} \rangle + \langle \gamma_t F(z_t), \hat{z}_t - z_{t+1} \rangle + \langle \gamma_t F(\hat{z}_t), z_{t+1} - z \rangle \\ &\leq \gamma_t \langle F(\hat{z}_t) - F(z_t), \hat{z}_t - z_{t+1} \rangle - V(z_{t+1}, \hat{z}_t) - V(\hat{z}_t, z_t) + V(z, z_t) - V(z, z_{t+1}) \end{aligned}$$

Let $\sigma_t = \gamma_t \langle F(\hat{z}_t) - F(z_t), \hat{z}_t - z_{t+1} \rangle - V(z_{t+1}, \hat{z}_t) - V(\hat{z}_t, z_t)$. By assumption of smoothness, we have $\|F(\hat{z}_t) - F(z_t)\|_* \leq L \|\hat{z}_t - z_t\|$. Invoking Cauchy-Schwatz inequality and the property of Bregman distance, $V(z, z') \geq \frac{1}{2} ||z - z'||^2$, to obtain

$$\sigma_t \leq \gamma_t L ||z_{t+1} - \hat{z}_t|| \cdot ||\hat{z}_t - z_t|| - \frac{1}{2} ||z_{t+1} - \hat{z}_t||^2 - \frac{1}{2} ||\hat{z}_t - z_t||^2.$$

Since $\gamma_t \leq 1/L$, we have $\sigma_t \leq 0$.

Thus we have

$$\langle \gamma_t F(\hat{z}_t), \hat{z}_t - z \rangle \leq V(z, z_t) - V(z, z_{t+1}).$$

Note that

$$\begin{aligned}\langle \gamma_t F(\hat{z}_t), \hat{z}_t - z\rangle &= \gamma_t[\langle \nabla_x \phi(\hat{x}_t, \hat{y}_t), \hat{x}_t - x\rangle + \langle -\nabla_y \phi(\hat{x}_t, \hat{y}_t), \hat{y}_t - y\rangle] \\ &\geq \gamma_t[\phi(\hat{x}_t, \hat{y}_t) - \phi(x, \hat{y}_t) + \phi(\hat{x}_t, y) - \phi(\hat{x}_t, \hat{y}_t)] \\ &= \gamma_t[\phi(\hat{x}_t, y) - \phi(x, \hat{y}_t)]\end{aligned}$$

where we have used that $\phi(x, \hat{y}_t)$ and $-\phi(\hat{x}_t, y)$ are convex and for any convex function f, $f(u) \geq f(v) + \langle \nabla f(v), u - v\rangle, \quad \forall u, v \in \mathrm{dom}(f)$.

Consider the sum up to T terms, and divide by $\sum_{t=1}^T \gamma_t$ to obtain

$$\frac{\sum_{t=1}^T \gamma_t[\phi(\hat{x}_t, y) - \phi(x, \hat{y}_t)]}{\sum_{t=1}^T \gamma_t} \leq \frac{V(z, z_1)}{\sum_{t=1}^T \gamma_t}.$$

Let $\bar{z}_T = (\bar{x}_T, \bar{y}_T) = \frac{\sum_{t=1}^T \gamma_t \hat{z}_t}{\sum_{t=1}^T \gamma_t}$, by convex-concavity of $\phi(x, y)$, this further implies,

$$\phi(\bar{x}_T, y) - \phi(x, \bar{y}_T) \leq \frac{V(z, z_1)}{\sum_{t=1}^T \gamma_t}, \quad \forall z = [x, y] \in X \times Y.$$

Taking the maximum over all $x \in X, y \in Y$, we obtain

$$\epsilon_{\mathrm{sad}}(\bar{z}_T) = \bar{\phi}(\bar{x}_T) - \underline{\phi}(\bar{y}_T) \leq \max_{z \in X \times Y} \frac{V(z, z_1)}{\sum_{t=1}^T \gamma_t} = \frac{\Omega}{\sum_{t=1}^T \gamma_t}.$$

■

Remark. If the step-size is assumed to be constant, $\gamma_t = \frac{1}{L}$, then we have

$$\epsilon_{\mathrm{sad}}(\bar{z}_T) \leq \frac{\Omega L}{T}.$$

Mirror Prox algorithm achieves a $O(1/T)$ rate of convergence using only first order information of $\phi(x, y)$.

Recall that

$$\epsilon_{\mathrm{sad}}(z) = \bar{\phi}(x) - \mathrm{Opt(P)} + \mathrm{Opt(D)} - \underline{\phi}(y),$$

Hence, both primal and dual error is bounded by $O(1/T)$. That is, when solving a nonsmooth convex minimization problem

$$\min_{x \in X} f(x), \text{ where } f(x) = \max_{y \in Y} \phi(x, y)$$

the Mirror Prox algorithm attains the $O(1/T)$ rate, comparable to Nesterov's smoothing technique. Note that this algorithm does not require f to be simple in order to allow for easy omputation of proximal operators of f but instead only requires operator F which comes from ϕ. In this regard, it is more general than Nesterov's smoothing technique.

Beyond saddle point problems. Mirror-prox algorithm can be used to solve a large range of problems for which we have the knowledge of operator F irrespective of whether it comes from gradient of a convex minimization problem, a saddle point problem or any other optimization problems. Indeed, this algorithm has been used widely to solve convex minimization, saddle point problems, variational inequalities, and fixed point problems.

References

[NEM '04] Arkadi Nemirovski, Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems, *SIAM Journal on Optimization*, 15(1):229-251, 2004.

[JN '11] Anatoli Juditsky, and Arkadi Nemirovski, First order methods for nonsmooth convex large-scale optimization, ii: utilizing problems structure, *Optimization for Machine Learning*, 149-183, 2011.

IE 598: Big Data Optimization **Fall 2016**

Lecture 19: Proximal Gradient Method and Its Acceleration

Lecturer: Niao He *Scriber: Lucas Buccafusca*

Overview: In this lecture, we introduce and analyze the Proximal Gradient method as a way of solving non-smooth convex problems with specific composite structure. We analyze the convergence rate and discuss the corresponding accelerated methods. Lastly, some simulations are done to demonstrate these algorithms for solving LASSO problem and compare algorithms we learned for nonsmooth optimization.

19.1 Motivations for Proximal Gradient Method

Our goal is to continue to address $\min_{x\in\mathbb{R}^n} f(x)$. Previously, we have discussed severak methods to solve it, based on the properties of f, assuming f is convex. For instance,

- f is smooth: Gradient Descent Method.

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t) = \operatorname*{argmin}_x \big\{ \frac{1}{2\gamma_t} \|x - [x_t - \gamma_t \nabla f(x_t)]\|_2^2 \big\}$$

- f is nonsmooth: Proximal Point Algorithm.

$$x_{t+1} = \text{prox}_{\gamma_t f}(x_t) = \operatorname*{argmin}_x \big\{ \frac{1}{2\gamma_t} ||x - x_t||_2^2 + f(x) \big\}$$

In this lecture, we will consider the special type of nonsmooth problems when the objective is the sum of a smooth and a nonsmooth function:

$$\min_x F(x) = \min_x \big\{ f(x) + g(x) \big\}$$

where $f(x)$ is a smooth and convex function and $g(x)$ is a non-smooth and convex function.

Note that neither gradient descent nor proximal point algorithm works well in this situation. We will introduce a new algorithm, **proximal gradient method**, which essentially combines gradient descent and proximal point algorithm and works as follows

$$x_{t+1} = \text{prox}_{\gamma_t g}(x_t - \gamma_t \nabla f(x_t)) = \operatorname*{argmin}_x \big\{ \frac{1}{2\gamma_t} ||x - [x_t - \gamma_t \nabla f(x_t)]||_2^2 + g(x) \big\}$$

We will discuss the details below.

19.2 Convex Composite Minimization

Problem setting. We consider the following convex minimization problem

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x)$$

where:

- $f(x)$ is L-smooth (i.e. gradient is Lipschitz continuous) and convex
- $g(x)$ is convex, 'simple' and non-smooth (By 'simple', we mean that the proximal operation of g is easy to calculate)

Applications. This type of optimization problem are found everywhere:

- Machine Learning: many supervised learning problems can be cast into optimization problems in the form
$$\min_{h\in\mathbb{H}} \sum_{i=1}^{n} l(h(x_i), y_i) + p(h)$$
where h is the prediction function to be learned, the loss function $\ell(\cdot,\cdot)$ is many cases are smooth functions, and the $p(h)$ is known as a penalty or regularization, which is often nonsmooth, e.g. L_1, L_2 or a combination of $L_1\&L_2$ norms. Specific examples of supervised learning include: ridge regression, LASSO, logisitic regression, etc.
- Signal and Image Processing: many signal/image recover problems can be formulated as the general form
$$\min_x ||Ax - b||_2^2 + r(x)$$
where the first term is the data fidelity term and the second term is some regularization. Such examples include compressive sensing, image deblurring, image denoising, etc.

Special cases: Note that the above problem covers many problems we have discussed as special cases

- nonsmooth convex minimization: when $f(x) = 0$
- smooth convex minimization : when $g(x) = 0$
- constrained convex minimizaton : when $g(x) = \delta_X(x)$ is an indicator function of a closed convex set X

19.3 Proximal Gradient Method

The proximal gradient (PG) method (dates back to [B75]) is fairly simple: at each iteration $t = 0, 1, 2, \ldots,$

$$x_{t+1} = \text{prox}_{\gamma_t g}(x_t - \gamma_t \nabla f(x_t)) = \underset{x}{\text{argmin}} \left\{ \frac{1}{2\gamma_t} ||x - [x_t - \gamma_t \nabla f(x_t)]||_2^2 + g(x) \right\} \tag{19.1}$$

Note that

- $f(x) = 0$: PG $\Rightarrow$ proximal point algorithm
- $g(x) = 0$: PG $\Rightarrow$ gradient descent
- $g(x) = \delta_X(x)$: PG $\Rightarrow$ projected gradient descent

LASSO Example of Proximal Gradient Method We provide a special example to help demonstrate how this algorithm works on the LASSO problem:

$$\min_x \underbrace{\frac{1}{2}||Ax-b||_2^2}_{f(x)} + \underbrace{\lambda||x||_1}_{g(x)}$$

First, we wish to calculate $\nabla f(x)$ and $\text{prox}_{\mu||\cdot||_1}(x)$. By inspection we have:

$$\nabla f(x) = A^T(Ax-b)$$
$$\text{prox}_{\mu||\cdot||_1}(x) = \underset{y}{\text{argmin}} \left\{ \frac{1}{2\mu}\sum_i (x_i - y_i)^2 + \sum_i |y_i| \right\}$$

We can separate this into each of the ith coordinates:

$$[\text{prox}_{\mu||\cdot||_1}(x)]_i = \underset{y_i}{\text{argmin}} \left\{ \tfrac{1}{2\mu}(x_i - y_i)^2 + |y_i| \right\}$$

The solution to this type of problem is well-known as the *soft thresholding operator*.

$$S_\mu(x_i) = \begin{cases} x_i - \mu & \text{if } x_i > \mu \\ 0 & \text{if } |x_i| \le \mu \\ x_i + \mu & \text{if } x_i < -\mu \end{cases}$$

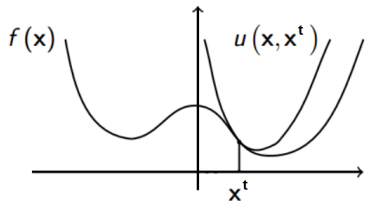
For each coordinates, we have the solution in the form of this soft thresholding operator, so we can rewrite our LASSO problem in Proximal Gradient form as

$$x_{t+1} = S_{\lambda\gamma_t}(x_t - A^T(Ax_t - b))$$

This is known as the Iterative Soft Thresholding Algorithm (ISTA).

Interpretation of Proximal Gradient Method. There are several ways in which we can interpret the Proximal Gradient Method:

1. *Majorization and Minimization:* A more general framework to solving minimization problems is to find an upper bound to your function and then try to minimize that bound. This framework is called majorization and minimization. This is done even in non-convex instances. Pictorally, we represent the Majorization and Minimization (otherwise known as MM) method below, where u(x) is the upper bound to our function [P16].



   Note that the updates (19.1) of proximal gradient method at each iteration can be rewritten as

$$x_{t+1} = \underset{x}{\text{argmin}} \left\{ f(x_t) + \nabla f(x_t)^T(x - x_t) + \frac{1}{2\gamma_t}||x - x_t||_2^2 + g(x) \right\}$$

If $\gamma_t \leq \frac{1}{L}$, then due to smoothness we have:

$$f(x_t) + \nabla f(x_t)^T(x - x_t) + \frac{1}{2\gamma_t}||x - x_t||_2^2 \geq f(x)$$

so we have an upper bound function that we minimize over, with equivalence at $x = x_t$.

2. *Fixed Point Iteration:* The proximal gradient method can also be treated as an iterative algorithm for a fixed point problem.

   Lemma 19.1 *x^* is optimal if and only if $\forall \gamma > 0$: $x^* = \text{prox}_{\gamma g}(x^* - \gamma \nabla f(x^*))$.*

   Proof: On the one hand, we have

   $$x^* \text{ is optimal } \Leftrightarrow 0 \in \nabla f(x^*) + \partial g(x^*).$$

   On the other hand, we have

   $$x^* = \text{prox}_{\gamma g}(x^* - \gamma \nabla f(x^*)) \Leftrightarrow 0 \in \frac{1}{\gamma}(x^* - (x^* - \gamma \nabla f(x^*))) + \partial g(x^*) \Leftrightarrow 0 \in \nabla f(x^*) + \partial g(x^*).$$

   ■

3. *Forward-Backward Operator:* We can equivalently write the proximal gradient operation as:

   $$x_{t+1} = (I + \gamma_t \partial g)^{-1}(I - \gamma_t \nabla f)(x_t)$$

   The $(I - \gamma_t \nabla f)$ is the 'forward' portion of the algorithm, e.g. a gradient step, and $(I + \gamma_t \partial g)^{-1}$ is the 'backward' portion of the algorithm, i.e. the proximal operator. This concept is typically used to solve general problems given by the sum of two maximal monotone operators.

19.4 Convergence Rate

Now we analyze the convergence of this algorithm. We show that just like gradient descent or proximal point algorithm, the proximal gradient method attains the $O(1/t)$ convergence rate.

Theorem 19.2 *Proximal gradient method with fixed step size $\gamma_t = \frac{1}{L}$ satisfies the following convergence rate:*

$$F(x_t) - F(x^*) \leq \frac{L||x_0 - x^*||_2^2}{2t}$$

Proof: For notation purposes, we can write: $x_{t+1} = x_t - \gamma_t G_{\gamma_t}(x_t)$ where $G_\gamma(x) = \frac{1}{\gamma}(x - \text{prox}_{\gamma g}(x - \gamma \nabla f(x)))$ From Lipschitz smoothness of $f(x)$, we know that the quadratic upper bound of any function is:

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{L}{2}||y - x||_2^2$$

Using the above property for $y = x - \gamma_t G_\gamma(x)$

$$f(x - \gamma_t G_\gamma(x)) \leq f(x) + \nabla f(x)^T((x - \gamma_t G_\gamma(x)) - x) + \tfrac{L}{2}||x - \gamma_t G_\gamma(x) - x||_2^2$$

$$\leq f(x) - \gamma_t \nabla f(x)^T G_\gamma(x) + \tfrac{L\gamma_t^2}{2}||G_\gamma(x)||_2^2$$

$$\leq f(x) - \gamma_t \nabla f(x)^T G_\gamma(x) + \tfrac{\gamma_t}{2}||G_\gamma(x)||_2^2 \text{ for } \gamma_t \leq \tfrac{1}{L}$$

Claim: For any y, we have

$$F(x - \gamma_t G_\gamma(x)) \leq F(y) + \nabla G_\gamma(x)^T (x - y) - \tfrac{\gamma_t}{2}||G_\gamma(x)||_2^2$$

Proof of Claim: From above, when applied to our function F:
$$F(x - \gamma_t G_\gamma(x)) \leq f(x) - \gamma_t \nabla f(x)^T G_\gamma(x) + \tfrac{\gamma_t}{2}||G_\gamma(x)||_2^2 + g(x - \gamma_t G_\gamma(x))$$

From the convexity of f and g and the fact that $G_\gamma(x) - \nabla f(x) \in \partial g(x - \gamma_t G_\gamma(x))$,

$$\begin{aligned} F(x - \gamma_t G_\gamma(x)) \leq \quad & f(y) + \nabla f(x)^T (x - y) - \gamma_t \nabla f(x)^T G_\gamma(x) + \tfrac{\gamma_t}{2}||G_\gamma(x)||_2^2 \\ & + g(y) + (G_\gamma(x) - \nabla f(x))^T (x - y - \gamma_t G_\gamma(x)) \end{aligned}$$

Simplifying and canceling terms nets:
$$F(x - \gamma_t G_\gamma(x)) \leq f(y) + g(y) + G_\gamma(x)^T (x - y) - \tfrac{\gamma_t}{2}||G_\gamma(x)||_2^2$$

Thus we have proved the Claim.

So now, applying the inequality f at $x = x_t$ an $y = x^*$, we have:

$$\begin{aligned} F(x_{t+1}) - F(x^*) &\leq G_{\gamma_t}(x_t)^T (x_t - x^*) - \frac{\gamma_t}{2}||G_{\gamma_t}(x_t)||_2^2 && (19.2) \\ &= \frac{1}{2\gamma_t}\left[||x_t - x^*||_2^2 - ||x_t - x^* - \gamma_t G_{\gamma_t}(x_t)||_2^2\right] && (19.3) \\ &= \frac{1}{2\gamma_t}\left[||x_t - x^*||_2^2 - ||x_{t+1} - x^*||_2^2\right] && (19.4) \end{aligned}$$

Take sums of both sides over all t and consider $\gamma_t = \frac{1}{L}$
$$\textstyle\sum_{i=1}^t (F(x_i) - F(x^*)) \leq \frac{L}{2} \sum_{i=1}^t \left[||x_i - x^*||_2^2 - ||x_{i+1} - x^*||_2^2\right] \leq \frac{L}{2}||x_0 - x^*||_2^2$$

Now we get:

$$F(x_t) - F(x^*) \leq \frac{1}{t}\sum_{i=1}^t (F(x_i) - F^*) \leq \frac{L}{2t}||x_0 - x^*||_2^2$$

■

Note that this convergence rate is the same as gradient descent and proximal point algorithm. Similarly, this is not the best rate achievable. We can also have accelerated proximal gradient descent methods.

19.5 Acceleration

19.5.1 Proximal Gradient with Backtracking Line-Search

Often times during our analysis, we set $\gamma_t = \frac{1}{L}$, but in practice, we do not know L a priori, or it is difficult to solve for. For example, for the LASSO problem, $L = \lambda_{max}(A^T A)$, but A may be large and difficult to work with. In practice we use backtracking line-search to find the local Lipschitz constant.

Backtracking line-search for Lipschitz constant Here is how the line-search works

- we initialize $L_0 = 1$ and some $\alpha > 1$.
- At each iteration t, we find the smallest integer i such that $L = \alpha^i L_{t-1}$ satisfies the Lipschitz condition, specifically:
$$F(x^+) \leq F(x_t) + \nabla f(x_t)(x^+ - x_t) + \tfrac{L}{2}||x^+ - x_t||_2^2$$
where $x^+ = \text{prox}_{\frac{g}{L}}(x_t - \frac{1}{L}\nabla f(x_t))$.
Then update $L_t = L$ and $x_{t+1} = x^+$

19.5.2 Accelerated Proximal Gradient Method

Originally developed by [Nesterov, 2007] and [Beck and Teboulle, 2009] , we can accelerate the proximal gradient method simply as follows

$$x_{t+1} = \text{prox}_{\gamma_t g}(y_t - \gamma_t \nabla f(y_t))$$

$$y_{t+1} = x_{t+1} + \beta_t(x_{t+1} - x_t)$$

Some simple choices for β: $\beta_t = \frac{t}{t+3}$ or $\beta_t = \frac{\lambda_t - 1}{\lambda_{t+1}}$ where $\lambda_0 = 0, \lambda_{t+1} = \frac{1+\sqrt{1+4\lambda_t^2}}{2}$.

The latter choice of acceleration is =called the Fast Iterative Soft Thresholding Algorithm (FISTA) derivation [BT09]. It was shown in [BT09] that

Theorem 19.3 *The sequences $x_t, F(x_t)$ generated via FISTA with either a constant or backtracking (with ratio $\alpha \geq 1$) stepsize rule satisfy*
$$F(x_t) - F(x^*) \leq \frac{2\alpha L}{t^2}||x_0 - x^*||_2^2$$

Remarks: By far, we have shown that when solving the composite convex minimization problem $\min_x \ f(x)+ g(x)$, we can the accelerated proximal algorithm attains the optimal $O(1/t^2)$ convergence rate.

19.6 Simulation

Some simulations of the LASSO problem were demonstrated in class [6].

$$\min_x \ \frac{1}{2}||Ax - b||_2^2 + \lambda||x||_1$$

We implement and compare all the algorithms we learned so far for solving nonsmooth problems, including

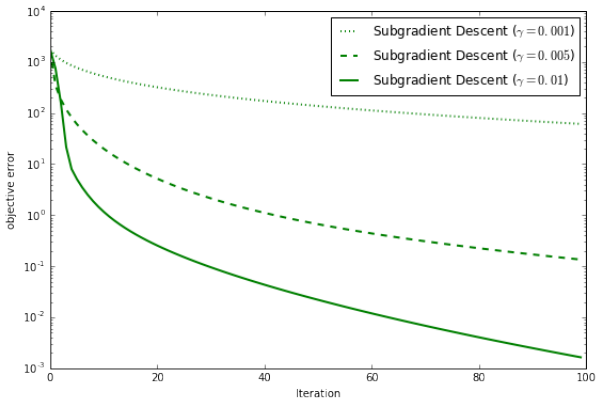
- Subgradient descent
- Nesterov's smoothing
- Proximal gradient (w/o backtracking)
- Accelerated proximal gradient (w/o backtracking)

We summarize the convergence rates of each of the five methods in the table below:

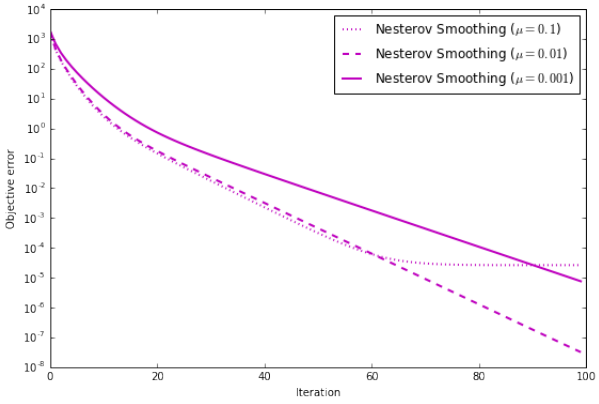
Method	Convergence	Parameter
Subgradient Descent	$O(\frac{1}{\sqrt{t}})$	stepsize $\gamma_t = \frac{\gamma}{\sqrt{t}}$
Nesterov Smoothing + GD	$O(\frac{1}{\mu t})$	smoothness $\mu > 0$
Nesterov Smoothing + AGD	$O(\frac{1}{\mu t^2})$	smoothness $\mu > 0$
Proximal Gradient Descent	$O(\frac{1}{t})$	stepsize $\gamma_t = \frac{1}{L}$ or line-search
Accelerated Proximal Gradient	$O(\frac{1}{t^2})$	stepsize $\gamma_t = \frac{1}{L}$ or line-search

We provide the results below:

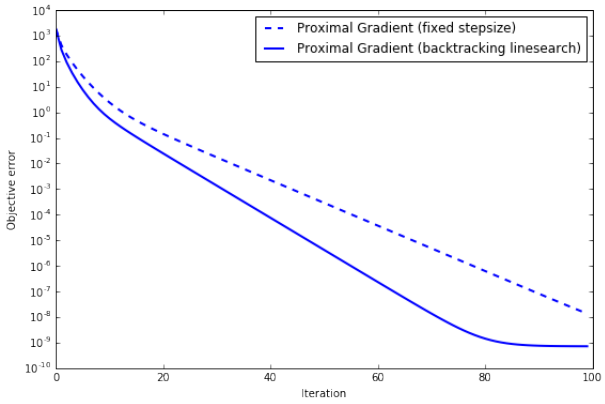
- Subgradient Descent - since the optimal stepsize is not computable, we set $\gamma_t = \frac{\gamma}{\sqrt{t}}$ and adjust γ.



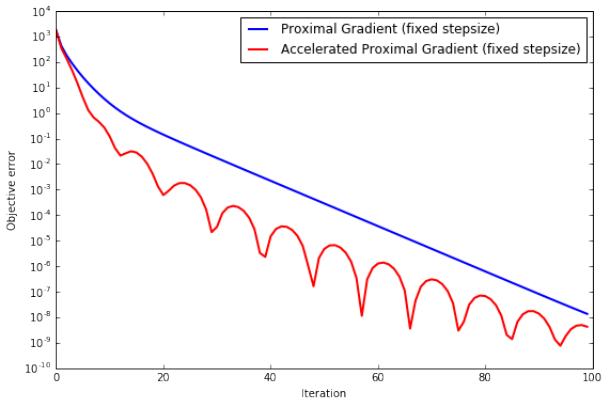
- Nesterov Smoothing - we approximate the nonsmooth L_1 norm by the Huber function and solve the resulting smoothed problem with Gradient Descent and adjust the smoothness parameter μ.



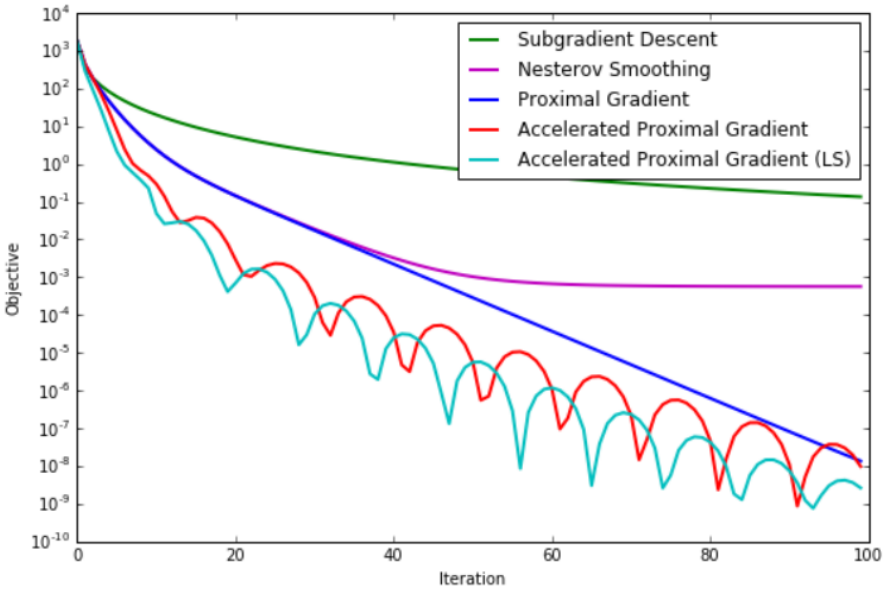
- Proximal Gradient - the only parameter here is the stepsize. We apply both fixed stepsize and backtracking line search. We see that backtracking helps.



- Accelerated Proximal Gradient - We can improve on Proximal Gradient Descent by applying the Accelerated method. With the same parameters of a fixed stepsize, we can immediately see the improvement.



Below is the overall comparison of the all five methods applied on the LASSO problem:



References

[B75] Bruck Jr, Ronald E., "An iterative solution of a variational inequality for certain monotone operators in Hilbert space." Bulletin of the American Mathematical Society 81, no. 5 (1975): 890-892.

[P16] Palomar, Daniel , "ELEC 5470-Convex Optimization Course Notes", 2016.

[CP09] Combettes, Patrick L.; Pesquet, Jean-Christophe, "Proximal Splitting Methods in Signal Processing." (2009) arXiv:0912.3522

[N07] Nesterov, Yu , "Gradient methods for minimizing composite objective function" No. CORE Discussion Papers (2007/76). UCL, 2007.

[BT09] Beck, A., and Teboulle, M. , "A fast iterative shrinkage-thresholding algorithm for linear inverse problems." SIAM journal on imaging sciences, 2(1), 183-202, 2009.

[H16] He, Niao , "Python demo for LASSO " `http://niaohe.ise.illinois.edu/IE598/lasso_demo/index.html`, 2016.

IE 598: Big Data Optimization **Fall 2016**

Lecture 20: Splitting Algorithms

Lecturer: Niao He *Scribers: Juho Kim*

In this lecture, we discuss splitting algorithms for convex minimization problems with objective given by the sum of two nonsmooth functions. We start with the fixed point property of such problems and derive a general scheme of splitting algorithm based on fixed point iteration. This covers Douglas-Rachford splitting and Peaceman-Rachford splitting algorithms. We also discuss the convergence rate of the KM algorithm for general fixed point problem.

20.1 Introduction: Proximal Algorithms

Previously we have considered several proximal algorithms for convex problems under different settings.

- **Nonsmooth Minimization:** $\min_{x\in\mathbb{R}^n} f(x)$, where $f(x)$ is convex and nonsmooth. We show the optimal solution has the following fixed point property:

$$x_* \text{ is optimal } \iff 0 \in \partial f(x_*) \iff \forall \lambda > 0,\ x_* = \text{prox}_{\lambda f}(x_*) \tag{20.1}$$

  The fixed point iteration gives rise to the *proximal point algorithm*:

$$x_{t+1} = \text{prox}_{\lambda_t f}(x_t)$$

- **Smooth + Nonsmooth Minimization:** $\min_{x\in\mathbb{R}^n} f(x) + g(x)$, where $f(x)$ is smooth and convex, and $g(x)$ is nonsmooth and convex. In this case,

$$x_* \text{ is optimal } \iff 0 \in \nabla f(x_*) + \partial g(x_*). \iff \forall \lambda > 0,\ x_* = \text{prox}_{\lambda g}(x_* - \lambda \nabla f(x_*)) \tag{20.2}$$

  The fixed point iteration gives rise to the *proximal gradient algorithm*:

$$x_{t+1} = \text{prox}_{\lambda g}(x_t - \lambda_t \nabla f(x_t))$$

In this lecture, we consider the problem

- **Nonsmooth + Nonsmooth Minimization:** $\min_{x\in\mathbb{R}^n} f(x) + g(x)$, where both $f(x)$ and $g(x)$ are nonsmooth and convex functions. Apparently, we still have the fixed point property:

$$x_* \text{ is optimal } \iff 0 \in \partial f(x_*) + \partial g(x_*) \iff \forall \lambda > 0,\ x_* = \text{prox}_{\lambda(f+g)}(x_*) \tag{20.3}$$

  The fixed point iteration leads to

$$x_{t+1} = \text{prox}_{\lambda_t(f+g)}(x_t)$$

However, this would require the proximal operator of the sum of two convex function, which is not always easy to compute, even if the proximal operators of both functions separately may be easy to compute. For example, let $f(x) = \|x\|_1$ and $g(x) = \|Ax\|_2^2$. The proximal operators of f, g are given by

$$\text{prox}_f(y) = \text{argmin}_x \left\{ \frac{1}{2}\|x - y\|_2^2 + \|x\|_1 \right\}$$

$$\text{prox}_g(y) = \text{argmin}_x \left\{ \frac{1}{2}\|x-y\|_2^2 + \|Ax\|_2^2 \right\}$$

both are easy to compute. However, the proximal operator of the sum of the two functions $f+g$ is given by

$$\text{prox}_{(f+g)}(x) = \text{argmin}_x \left\{ \frac{1}{2}\|x-y\|_2^2 + \|x\|_1 + \|Ax\|_2^2 \right\}$$

is not easy to calculate.

Therefore, we need the above fixed point property (20.4) is not really useful. Intuitively, one might guess that $x_* = \text{prox}_{\lambda f}\left(\text{prox}_{\lambda g}(x)\right)$ and one can update $x_{t+1} = \text{prox}_{\lambda f}\left(\text{prox}_{\lambda g}(x_t)\right)$ by alternatively computing the proximal operators. For instance, if $f = \delta_{X_1}(\cdot)$ and $g = \delta_{X_2}(\cdot)$ are two indicator functions of convex sets X_1, X_2, this would imply an alternative projection on set X_1 and X_2. But this is not always true.

We show that instead, we have the following fixed point property:

$$x_* \text{ is optimal } \iff 0 \in \partial f(x_*) + \partial g(x_*) \iff \forall \lambda > 0,\ x_* = \text{prox}_{\lambda f}(y_*) \text{ and } y_* = \text{refl}_{\lambda g}(\text{refl}_{\lambda f}(y_*)) \quad (20.4)$$

where $\text{refl}_f = 2\text{prox}_f(x) - x$ is the reflection of the proximal operator. The corresponding fixed point iteration then leads to the so-called **splitting algorithms**. We now discuss the details.

20.2 Fix Point Theorem for Nonsmooth + Nonsmooth Problems

Consider the convex optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + g(x) \quad (20.5)$$

where both $f(x)$ and $g(x)$ are proper convex (perhaps nonsmooth) functions.

Lemma 20.1 *x_* is optimal to (20.5) if and only if for any $\lambda > 0$ and $\rho \in \mathbb{R}$,*

$$x_* = prox_{\lambda f}(y_*) \text{ and } y_* = y_* + \rho[prox_{\lambda g}(2prox_{\lambda f}(y_*) - y_*) - prox_{\lambda f}(y_*)] \quad (20.6)$$

Proof: Suppose that x_* is optimal.

$$\begin{aligned}
x_* \text{ is optimal } &\iff 0 \in \partial f(x_*) + \partial g(x_*) \\
&\iff \forall \lambda > 0, \text{there exists } z \text{ such that } z \in \partial(\lambda f)(x_*) \text{ and } -z \in \partial(\lambda g)(x_*) \\
&\iff \forall \lambda > 0, \text{there exists } y \text{ such that } y - x_* \in \partial(\lambda f)(x_*) \text{ and } x_* - y \in \partial(\lambda g)(x_*) \\
&\iff \forall \lambda > 0, \text{there exists } y \text{ such that } y - x_* \in \partial(\lambda f)(x_*) \text{ and } 2x_* - y \in x_* + \partial(\lambda g)(x_*) \\
&\iff x_* = \text{prox}_{\lambda f}(y) \text{ and } 2\text{prox}_{\lambda f}(y) - y \in x_* + \partial(\lambda g)(x_*) \\
&\iff x_* = \text{prox}_{\lambda f}(y) \text{ and } x_* = \text{prox}_{\lambda g}(2\text{prox}_{\lambda f}(y) - y) \\
&\iff x_* = \text{prox}_{\lambda f}(y) \text{ and } y = y + \rho[\text{prox}_{\lambda g}(2\text{prox}_{\lambda f}(y) - y) - \text{prox}_{\lambda f}(y)], \forall \rho
\end{aligned}$$

Thus, both statements are equivalent. ■

Remark

1. when $\rho = 1$: we have
$$y_* = \frac{1}{2}[y_* + 2\text{prox}_{\lambda g}(2\text{prox}_{\lambda f}(y_*) - y_*) - (2\text{prox}_{\lambda f}(y_*) - y_*)] = \frac{1}{2}[y_* + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y_*)].$$
Hence, $y_* = \mathcal{T}_{\lambda,f,g}(y_*)$ with the operator
$$\mathcal{T}_{\lambda,f,g} = \frac{1}{2}[I + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}]$$
this is known as the **Douglas-Rachford operator** [Lions and Mercier, 1979].

2. when $\rho = 2$: we have
$$y_* = y_* + 2\text{prox}_{\lambda g}(2\text{prox}_{\lambda f}(y_*) - y_*) - (2\text{prox}_{\lambda f}(y_*) - y_*) = y_* + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y_*).$$
Hence, $y_* = \mathcal{T}_{\lambda,f,g}(y_*)$ with the operator
$$\mathcal{T}_{\lambda,f,g} = \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}$$
this is known as the **Peaceman-Rachford operator** [Lions and Mercier, 1979].

Previously in Lecture 17, we show that the proximal operator is firmly nonexpansive, i.e.,
$$\|\text{prox}_f(x) - \text{prox}_f(y)\|_2^2 \leq \langle \text{prox}_f(x) - \text{prox}_f(y), x - y\rangle.$$
Indeed, both the Douglas-Rachford operator and the Peaceman-Rachford operator are both non-expansive operators, i.e. $\|\mathcal{T}_{\lambda,f,g}(x) - \mathcal{T}_{\lambda,f,g}(y)\|_2 \leq \|x - y\|_2, \forall x, y$.

First, we see that

Lemma 20.2 *The reflection operator $\text{refl}_{\lambda f}(\cdot)$ is non-expansive for any $\lambda > 0$.*

Proof: This is because
$$\begin{aligned}
\|\text{refl}_{\lambda f}(x) - \text{refl}_{\lambda f}(y)\|_2^2 &= \|2\text{prox}_{\lambda f}(x) - 2\text{prox}_{\lambda f}(y) - (x - y)\|_2^2 \\
&= 4\|\text{prox}_{\lambda f}(x) - \text{prox}_{\lambda f}(y)\|_2^2 - 4\langle \text{prox}_{\lambda f}(x) - \text{prox}_{\lambda f}(y), x - y\rangle + \|x - y\|_2^2 \\
&\leq 4\|\text{prox}_{\lambda f}(x) - \text{prox}_{\lambda f}(y)\|_2^2 - 4\|\text{prox}_{\lambda f}(x) - \text{prox}_{\lambda f}(y)\|_2^2 + \|x - y\|_2^2 \\
&= \|x - y\|_2^2
\end{aligned}$$
■

We are now able to show that

Lemma 20.3 *(i) The Peaceman-Rachford operator $T_1 = \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}$ is non-expansive.*

(ii) The Douglas-Rachford operator $T_2 = \frac{1}{2}[I + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}]$ is non-expansive.

Proof:
$$\|T_1 x - T_1 y\|_2 = \|\text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(x) - \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y)\|_2 \leq \|\text{refl}_{\lambda f}(x) - \text{refl}_{\lambda f}(y)\|_2 \leq \|x - y\|_2$$
where the first inequality is due to the non-expansiveness of $\text{prox}_{\lambda g}$ and the second inequality is due to the non-expansiveness of $\text{prox}_{\lambda f}$. Since $T_2 = \frac{1}{2}[I + T_1]$, then
$$\|T_2 x - T_2 y\|_2 \leq \frac{1}{2}\|x - y\|_2 + \frac{1}{2}\|T_1 x - T_1 y\|_2 \leq \|x - y\|_2$$
Thus, we have shown that both operators are non-expansive. ■

20.3 Splitting algorithms

The fixed point iterations corresponding to these non-expansive operators lead to the following algorithms:

1. *Douglas-Rachford splitting algorithm:*

$$y_{t+1} = \frac{1}{2}[y_t + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y_t)]$$

2. *Peaceman-Rachford splitting algorithm:*

$$y_{t+1} = \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y_t)$$

3. *Relaxed Peaceman-Rachford splitting algorithm:* let $\gamma_t \in (0, 1]$,

$$y_{t+1} = (1 - \gamma_t)y_t + \gamma_t \cdot \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(y_t)$$

   (Special cases) $\gamma_t = 1$: Peaceman-Rachford splitting and $\gamma_t = \frac{1}{2}$: Douglas-Rachford splitting.

In the following, we illustrate the details of the Douglas-Rachford splitting algorithm and demonstrate that indeed it is a special case of the well-known Alternating Direction Method of Multipliers (ADMM).

20.3.1 Douglas-Rachford splitting

Let us initialize y_1 and $x_1 = \text{prox}_{\lambda f}(y_1)$, the Douglas-Rachford splitting algorithm can be rewritten as

$$\begin{cases} y_{t+1} = y_t + \text{prox}_{\lambda g}(2x_t - y_t) - x_t \\ x_{t+1} = \text{prox}_{\lambda f}(y_t) \end{cases}$$

Let $z_{t+1} = \text{prox}_{\lambda g}(2x_t - y_t)$, this can be further formulated as

$$\begin{cases} z_{t+1} = \text{prox}_{\lambda g}(2x_t - y_t) \\ x_{t+1} = \text{prox}_{\lambda f}(y_t + z_{t+1} - x_t) \\ y_{t+1} = y_t + z_{t+1} - x_t \end{cases}$$

Let $u_t = x_t - y_t$.

$$\begin{cases} z_{t+1} = \text{prox}_{\lambda g}(x_t + u_t) \\ x_{t+1} = \text{prox}_{\lambda f}(z_{t+1} - u_t) \\ u_{t+1} = u_t + (x_{t+1} - z_{t+1}) \end{cases}$$

which is a special case of the Alternating Direction Methods of Multipliers (ADMM).

20.3.2 Alternating Direction Methods of Multipliers (ADMM)

We consider the following optimization problem.

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c \end{aligned} \tag{20.7}$$

Let $\rho > 0$, the augmented Lagrangian for this optimization is given as follows.

$$L_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2.$$

ADMM consists of the iterations.

$$\begin{cases} z_{t+1} = \text{argmin}_z L_\rho(x_t, z, \lambda_t) \\ x_{t+1} = \text{argmin}_x L_\rho(x, z_{t+1}, \lambda_t) \\ \lambda_{t+1} = \lambda_t + \rho(Ax_{t+1} + Bz_{t+1} - c) \end{cases}$$

Letting $u_t = \frac{\lambda_t}{\rho}$, the iterations above are the followings

$$\begin{cases} z_{t+1} = \text{argmin}_z\{g(z) + \frac{\rho}{2}\|Ax_t + Bz - c + u_t\|_2^2\} \\ x_{t+1} = \text{argmin}_x\{f(x) + \frac{\rho}{2}\|Ax + Bz_{t+1} - c + u_t\|_2^2\} \\ u_{t+1} = u_t + (Ax_{t+1} + Bz_{t+1} - c) \end{cases}$$

A general convex optimization problem such as $\min_x f(x) + g(x)$ can be converted as the form that ADMM can be applied to.

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{s.t.} \quad & x - z = 0 \end{aligned}$$

This is a special case of (20.7) with $A = I, B = -I, c = 0$. Let $\rho = \frac{1}{\lambda}$, one can see that the ADMM algorithm is exactly the same as the Douglas-Rachford splitting algorithm in this case.

20.4 Convergence Analysis

Here we provide a unified analysis for fixed point iterative algorithms.

Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be a nonexpansive operator. The goal is to find a fixed point x_* such that $x_* = Tx_*$. We consider the relaxed fixed point algorithm

$$x_{t+1} = (1 - \gamma_t)x_t + \gamma_t \cdot Tx_t, \ \textit{for all } t \geq 0$$

where $\gamma_t \in (0, 1]$. This is known as the **Krasnosel'skii-Mann(KM) algorithm**. Note that when $\gamma_t = 1$, this reduces to the usual fixed point algorithm.

We point out that this algorithm covers most of the algorithms we discussed so far,

- Gradient descent: $x_{t+1} = Tx_t$, where $Tx = x - \frac{1}{L}\nabla f(x)$
- Projected gradient descent: $x_{t+1} = Tx_t$, where $Tx = \Pi_X(x - \frac{1}{L}\nabla f(x))$
- Proximal gradient descent: $x_{t+1} = Tx_t$, where $Tx = \text{prox}_{\frac{g}{L}}(x - \frac{1}{L}\nabla f(x))$
- Proximal point algorithm: $x_{t+1} = Tx_t$, where $Tx = \text{prox}_{\lambda f}(x)$
- Douglas-Rachford algorihtm : $x_{t+1} = Tx_t$, where $Tx = \frac{1}{2}[y + \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(x)]$
- Relaxed Peaceman-Rachford algorihtm : $x_{t+1} = (1 - \gamma_t)x_t + \gamma_t \cdot Tx_t$, where $Tx = \text{refl}_{\lambda g} \circ \text{refl}_{\lambda f}(x)$

Theorem 20.4 *Let T be a nonexpansive operator. The KM algorithm satisfies that*

$$\|Tx_t - x_t\|_2^2 \leq \frac{\|x_0 - x_*\|_2^2}{\sum_{\tau=0}^{t} \gamma_\tau(1-\gamma_\tau)}.$$

Proof: We first show that $\|x_t - Tx_t\|_2$ is non-increasing.

$$\begin{aligned}
\|x_{t+1} - Tx_{t+1}\|_2 &= \|(1-\gamma_t)x_t + \gamma_t Tx_t - Tx_{t+1}\|_2 \\
&= \|(1-\gamma_t)(x_t - Tx_t) + Tx_t - Tx_{t+1}\|_2 \\
&\leq (1-\gamma_t)\|x_t - Tx_t\|_2 + \|Tx_t - Tx_{t+1}\|_2 \quad \text{(Triangular inequality)} \\
&\leq (1-\gamma_t)\|x_t - Tx_t\|_2 + \|x_t - x_{t+1}\|_2 \quad \text{(T is a nonexpansive operator)} \\
&= (1-\gamma_t)\|x_t - Tx_t\|_2 + \gamma_t\|x_t - Tx_t\|_2 \\
&= \|x_t - Tx_t\|_2
\end{aligned}$$

We now show that

$$\begin{aligned}
\|x_{t+1} - x_*\|_2^2 &= \|(1-\gamma_t)x_t + \gamma_t Tx_t - (1-\gamma_t)x_* - \gamma_t Tx_*\|_2^2 \\
&= \|(1-\gamma_t)(x_t - x_*) + \gamma_t(Tx_t - Tx_*)\|_2^2 \\
&= (1-\gamma_t)\|x_t - x_*\|_2^2 + \gamma_t\|Tx_t - Tx_*\|_2^2 - \gamma_t(1-\gamma_t)\|x_t - Tx_t\|_2^2
\end{aligned}$$

The last equality is due to the fact that for any $\gamma \in [0,1]$ and u, v,

$$\|(1-\gamma)u + \gamma v\|_2^2 = (1-\gamma)\|u\|_2^2 + \gamma\|v\|_2^2 - \gamma(1-\gamma)\|u - v\|_2^2$$

Therefore,

$$\|x_{t+1} - x_*\|_2^2 \leq \|x_t - x_*\|_2^2 - \gamma_t(1-\gamma_t)\|x_t - Tx_t\|_2^2$$

Taking summation over t leads to

$$\left(\sum_{\tau=0}^{t} \gamma_\tau(1-\gamma_\tau)\right) \cdot \|x_t - Tx_t\|_2^2 \leq \sum_{\tau=0}^{t} \gamma_\tau(1-\gamma_\tau)\|x_\tau - Tx_\tau\|_2^2 \leq \|x_0 - x_*\|_2^2.$$

Thus,

$$\|Tx_t - x_t\|_2^2 \leq \frac{\|x_0 - x_*\|_2^2}{\sum_{\tau=0}^{t} \gamma_\tau(1-\gamma_\tau)}.$$

■

Remark. In particular, if we set $\gamma_t = \gamma \in (0,1)$, we have

$$\|Tx_t - x_t\|_2^2 \leq \frac{\|x_0 - x_*\|_2^2}{\gamma(1-\gamma)(t+1)}.$$

Thus, this indicates that the KM algorithm achieves an overall $O(1/t)$ rate of convergence for solving a fixed point problem. As an immediate fact, the relaxed Peaceman-Rachford algorithm also attains the same $O(1/t)$ rate. It remains interesting to investigate the acceleration of such algorithms; we will leave out the details here.

20.5 Example: LASSO

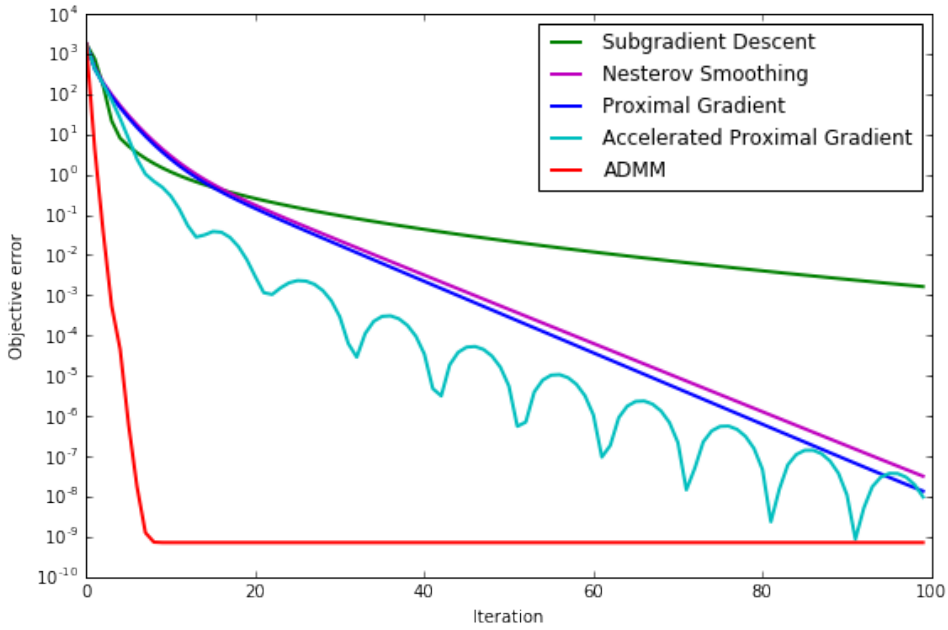
We illustrate the Douglas-Rachford/ADMM algorithm for solving the LASSO problem:

$$\min_x \underbrace{\frac{1}{2}||Ax-b||_2^2}_{f(x)} + \underbrace{\lambda||x||_1}_{g(x)}$$

The ADMM algorithm works as follows:

$$\begin{cases} z_{t+1} = S_{\lambda/\rho}(x_t + y_t/\rho) \\ x_{t+1} = (A^TA + \rho I)^{-1}(A^Tb + \rho z_{t+1} - \lambda_t) \\ \lambda_{t+1} = \lambda_t + \rho(x_{t+1} - z_{t+1}) \end{cases}$$

Below is the overall comparison of ADMM and the other five methods we implemented on the LASSO problem in the previous lecture:



We observe that ADMM significantly outperforms all other algorithms. One should note that in this case, ADMM treats the first data-fidelity term $f(x) = \frac{1}{2}\|Ax-b\|_2^2$ as a nonsmooth term and compute exactly its proximal operator at each iteration (which requires computing the inverse of a matrix), while other algorithms simply treat this as a smooth term and use only its gradient information at each iteration (which requires much cheaper computation cost).

References

[BPCPE11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, 2011.

[PB14] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends in Optimization*, 2014.

[RSV14] R. Cominetti, Roberto, J.A. Soto, and J. Vaisman, "On the rate of convergence of Krasnosel?ski?-Mann iterations and their connection with sums of Bernoullis." *Israel Journal of Mathematics* 199.2: 757-772, 2014.

[LM79] P.L. Lions, and B. Mercier, " Splitting algorithms for the sum of two nonlinear operators". *SIAM Journal on Numerical Analysis*, 16(6), 964-979, 1979.

[V16] L. Vandenberghe " Lecture Notes for EE236C: Optimization Methods for Large-Scale Systems ", Spring 2016.

[N16] He, Niao , "Python demo for LASSO " `http://niaohe.ise.illinois.edu/IE598/lasso_demo/index.html`, 2016.

IE 598: Big Data Optimization **Fall 2016**

Lecture 21: Stochastic Optimization – November 3

Lecturer: Niao He *Scriber: Juan Xu*

Overview In this lecture, we consider a new type of optimization problem involving uncertainty, which is called stochastic optimization. We introduce two approaches, sample average approximation and stochastic approximation, to solve stochastic optimization problems.

21.1 Introduction

When modeling and solving a realistic optimization problem, we always face uncertainty presenting in the problem, e.g., unknown parameters. The uncertainty makes both the realism and tractability of optimization problems more difficult. In this section, we introduce the formulation with uncertainty and some examples stimulating from real-life decision making request to give a good description of stochastic optimization.

Stochastic Optimization Formulation

The stochastic optimization problem is often formulated as the following format

$$\min_{x\in\mathbf{X}} f(x) = \mathbb{E}_\xi \left[F(x,\xi)\right], \tag{P}$$

where $F(x,\xi)$ is a function involving our decision variable (vector) x and a random variable (vector) ξ. The random ξ is some well-defined random variable with support $\Omega \subseteq \mathbb{R}^d$ and a distribution P. We minimize $f(x)$ to get the optimization solution, where the function $f(x)$ without uncertainty is given by taking the expectation of $F(x,\xi)$ over ξ, i.e.,

$$\mathbb{E}_\xi \left[F(x,\xi)\right] = \int_{\xi\in\Omega} F(x,\xi) dP(\xi).$$

Note that $F(x,\xi)$ is convex for any $\xi \in \Omega$, and by the calculus of convex functions, it can imply that $f(x)$ is convex. However, vice versa is not true.

Example 1. (Newsvendor model)

A newspaper vendor needs to decide how many copies of today's newspaper to stock in order to meet the uncertain demand and to maximize profit meanwhile. Suppose the number of newspaper to be stocked q is the vendor's decision variable, the purchase price per newspaper that the vendor needs to pay is denoted by c, the selling price per newspaper that consumers need to pay is denoted by p. We use D to represent the consumers' random demand for newspaper. Then, the Newsvendor model can be formulated as

$$\max_q \mathbb{E}_D \left[p \cdot \min(q, D) - c \cdot q\right].$$

The cost is $c \cdot q$ which the vendor needs to pay for the stocked q copies of newspaper. The revenue the vendor can get is $p \cdot \min(q, D)$ which means the vendor cannot sell more than the stocked number of newspaper and the consumers' demand.

Note that the maximization problem is equivalent to the following minimization problem by adding a minus sign to the objection function, so it is consistent with the stochastic optimization formulation,

$$\min_q -\mathbb{E}_D \left[p \cdot \min(q, D) - c \cdot q \right].$$

Also note that the objective function of the minimization problem $-\mathbb{E}_D \left[p \cdot \min(q, D) - c \cdot q \right]$ is a convex function now, and the values of the maximization and minimization problem are opposite each other.

Example 2. (Markowitz model)

Suppose an investor wants to invest in different stocks with random returns in order to maximize the returns generating by buying these stocks, and to minimize the variance of the random returns meanwhile. We use $\mathbf{w}$ to denote the weights of stock and take it as our decision, and use $\mathbf{r}$ to denote the random returns. Then the Markowitz model is

$$\max_{\mathbf{w} \geq \mathbf{0}, \sum w_i = 1} \mathbb{E}_{\mathbf{r}} \left[\mathbf{w}^\top \cdot \mathbf{r} \right] - \lambda \cdot \mathrm{Var} \left[\mathbf{w}^\top \cdot \mathbf{r} \right],$$

where λ is a parameter used to penalize the variance of the random returns. We again translate the maximization problem into a minimization problem by adding a minus sign before the objective function, and it can be show that $-\mathbb{E}_{\mathbf{r}} \left[\mathbf{w}^\top \cdot \mathbf{r} \right] + \lambda \cdot \mathrm{Var} \left[\mathbf{w}^\top \cdot \mathbf{r} \right]$ is a convex function.

Example 3. (Expected risk minimization)

In many machine learning problems, we hope to minimize the expected loss given by the loss function $l(f(x), y)$ through choosing a suitable function f from the function set $\mathcal{F}$ where x and y are random input data. The formulation is

$$\min_{f \in \mathcal{F}} \mathbb{E}_{x,y} \left[l(f(x), y) \right].$$

Notice that the three examples are all stochastic optimization problems. A question appears: how to solve stochastic optimization problems? It is difficult to use the methods we have learned for deterministic problems here because it often is intractable to compute the gradient of $f(x)$ which involves an integration. Suppose $F(x, \xi)$ is differential for any $\xi \in \Omega$, the gradient $\nabla f(x) = \int_{\xi \in \Omega} \nabla F(x, \xi) dP(\xi)$, can be difficult to compute.

In this lecture, we introduce two approaches

1. sample average approximation
2. stochastic approximation

which can be used to solve stochastic optimization problems.

21.2 Sample Average Approximation

A natural way to address the stochastic optimization problem, is to use Monte Carlo sampling. Let $\xi_1, \ldots, \xi_N$ be independently and identically distributed (i.i.d.) random sample of the random variable (vector) ξ. We

consider the following estimation of the original problem

$$\min_{x\in\mathbf{X}} f^N(x) = \frac{1}{N}\sum_{i=1}^{N} F(x,\xi_i). \tag{SAA}$$

This i known as the **sample average approximation**.

For simplicity, we assume the feasible set $\mathbf{X}$ is a finite, then original stochastic optimization (P) and its sample average approximation (SAA) have nonempty sets of optimal solutions, denoted by $\mathbf{X}_*$ and $\mathbf{X}_*^N$ respectively. For each element in $\mathbf{X}_*$ and $\mathbf{X}_*^N$, it gives the optimal value of (P) and (SAA) respectively, i.e.,

$$\begin{aligned} f_* &= f(x_*) = \min_{x\in\mathbf{X}} f(x), \quad \forall x_* \in \mathbf{X}_*, \\ f_*^N &= f(x_*^N) = \min_{x\in\mathbf{X}} f^N(x), \quad \forall x_*^N \in \mathbf{X}_*^N. \end{aligned}$$

We define sets of ϵ-optimal solution set $\mathbf{X}_\epsilon$ and $\mathbf{X}_\epsilon^N$ for (P) and (SAA), respectively. That is,

$$\begin{aligned} &\forall \epsilon \geq 0, \quad \mathbf{X}_\epsilon := \{x \in \mathbf{X} : f(x) \leq f_* + \epsilon\}, \\ &\forall \epsilon \geq 0, \quad \mathbf{X}_\epsilon^N := \{x \in \mathbf{X} : f^N(x) \leq f_*^N + \epsilon\}. \end{aligned}$$

An extreme case is that when $\epsilon = 0$, set $\mathbf{X}_\epsilon$ coincides $\mathbf{X}_*$, and $\mathbf{X}_\epsilon^N$ coincides $\mathbf{X}_*^N$.

Proposition 21.1 *The following two properties hold:*

(1) $f_*^N \to f_*$ *w.p.1 as* $N \to \infty$*, and*

(2) $\forall \epsilon \geq 0, \mathbb{P}(\mathbf{X}_\epsilon^N \subseteq \mathbf{X}_\epsilon) = 1$ *as* $N \to \infty$*.*

Proof:

(1) Following from the SLLN (strong law of large numbers), $\forall x \in \mathbf{X}, f^N(x)$ converges to $f(x)$ w.p.1 as $N \to \infty$. In other words, $\forall x \in \mathbf{X}, |f^N(x) - f(x)| \to 0$ w.p.1. as $N \to \infty$. This means as $N \to \infty$, the set $\{x \in \mathbf{X} : |f^N(x) - f(x)| > 0\}$ has measure zero. Since set $\mathbf{X}$ is finite, and the union of a finite number of sets each of which has measure zero shows a measure of zero, we have $\delta_N := \max_{x\in\mathbf{X}} |f^N(x) - f(x)| \to 0$ w.p.1 as $N \to \infty$. Note that

$$|f_*^N - f_*| \leq \delta_N,$$

This is because

$$f_*^N - f_* = f^N(x_*^N) - f^N(x_*) + f^N(x_*) - f(x_*) \leq f^N(x_*) - f(x_*) \leq \delta_N \tag{21.1}$$

$$f_* - f_*^N = f(x_*) - f(x_*^N) + f(x_*^N) - f^N(x_*^N) \leq f(x_*^N) - f^N(x_*^N) \leq \delta_N \tag{21.2}$$

Hence, we have $|f_*^N - f_*| \to 0$, i.e. $f_*^N \to f_*$ w.p.1 as $N \to \infty$.

(2) For a given $\epsilon \geq 0$, we define

$$\rho(\epsilon) := \min_{x\in\mathbf{X}\setminus\mathbf{X}^\epsilon} f(x) - f_* - \epsilon.$$

By the definition of ϵ-optimal solution set $\mathbf{X}_\epsilon$ for (P), we have $\forall x \in \mathbf{X} \setminus \mathbf{X}_\epsilon, f(x) > f_* + \epsilon$. Because $\mathbf{X}$ is finite, then $\rho(\epsilon) > 0$.

Let N be large enough such that $\delta_N < \rho(\epsilon)/2$. Hence, $|f_*^N - f_*| \leq \delta_N < \rho(\epsilon)/2$, and furthermore, $f_*^N < f_* + \rho(\epsilon)/2$. In addition, we have $\forall x \in \mathbf{X} \setminus \mathbf{X}_\epsilon, |f^N(x) - f(x)| \leq \delta_N < \rho(\epsilon)/2$, which gives us $f^N(x) > f(x) - \rho(\epsilon)/2 \geq f_* + \epsilon + \rho(\epsilon)/2$. Combining the results above, it follows that $\forall x \in \mathbf{X} \setminus \mathbf{X}_\epsilon, f^N(x) > f_*^N + \epsilon$ and therefore x does not belong to the ϵ-optimal set $\mathbf{X}_\epsilon^N$ for (SAA). In other words, it tells us for $\forall \epsilon \geq 0, \mathbb{P}(\mathbf{X}_\epsilon^N \subseteq \mathbf{X}_\epsilon) = 1$ as $N \to \infty$.

■

This proposition tells us as $N \to \infty$, we can get a near-zero gap approximation for the optimal value f_* of (P) by the SAA formulation. What's more, as $N \to \infty$, for any $\epsilon \geq 0$, the ϵ-optimal solution set of (SAA) always lies in the ϵ-optimal solution set of (P).

Theorem 21.2 [*Kleywegt, Shapiro, Homem-De-Mellow*, 2001]
For $0 \leq \delta < \epsilon$, and integer N, we have

$$\mathbb{P}(\mathbf{X}_\delta^N \subseteq \mathbf{X}_\epsilon) \geq 1 - |\mathbf{X}| e^{-N \cdot \gamma(\delta, \epsilon)},$$

where $\gamma(\delta, \epsilon) \geq \frac{(\epsilon - \delta)^2}{4\sigma_{max}^2}$, and $\sigma_{max}^2 = \max\limits_{x \in \mathbf{X}} Var[F(x, \xi)]$.

Remark 1. When $\delta = 0$, for the optimal solution of the SAA problem to be an ϵ-optimal solution to (P) with probability $1 - \alpha$, the sample size N needs to be

$$N \geq \frac{4\sigma_{\max}^2}{\epsilon^2} \log\big(\frac{|\mathbf{X}|}{\alpha}\big).$$

We can see that even if the size of X increases exponentially, then lower bound of needed sample size N increases linearly. Also, the complexity depends linearly in the variance of $F(x, \xi)$. Overall, the sample complexity is of order $O(1/\epsilon^2)$.

Pros and Cons of SAA.

(+) : SAA is very general. As we see, SAA method doesn't make any assumptions about the convexity of our objective functions. Then even if our objective is a non-convex or even comes from discrete optimization, we can still use SAA method to solve the stochastic optimization problem, and the related results still hold. In addition, we can combine any existing algorithms to solve SAA problems.

(−) : Since the variance $\sigma_{\max}^2$ and $|X|$ is often unknown or hard to compute, in practice, it is difficult to detterminie an appropriate sample size N we need.

(−) : On the one hand, large sample size N can help us to get a high accuracy, while on the other hand, solving (SAA) problem with large N can be expensive. For instance, computing the gradient $\nabla f^N(x)$ requires to compute the sum of $O(N)$ gradients.

(−) : The SAA approach only works with batch data, and cannot handle streaming/online data.

21.3 Stochastic Approximation

Stochastic Approximation (SA) is another popular method to solve the stochastic optimization problem and it dates back to 1951 by Robbins and Monro [RM51]. Assume $F(x, \xi)$ is differential with x for any $\xi \in \Omega$, the idea of Classic Stochastic Approximation is that give a sample realization ξ_t, we update the decision variable (vector) x_{t+1} at iteration $t + 1$ following the rule:

$$x_{t+1} = \Pi_{\mathbf{X}}\big(x_t - \gamma_t \nabla F(x_t, \xi_t)\big),$$

which is also known as the stochastic gradient descent (SGD) method.

Remark 2.

1. The stochastic gradient is unbiased, i.e., $\mathbb{E}\left[\nabla F(x,\xi)\right] = \nabla f(x)$.

2. We need a decreasing sequence $\{\gamma_t\}$ and $\gamma_t \to 0$ as t goes to infinity to ensure convergence. Because at optimality, we have $x_* = x_* - \gamma \nabla F(x_*, \xi)$. However, since $\nabla f(x_*, \xi)$ is random, we cannot guarantee that $\nabla F(x_*, \xi) = 0, \forall \xi \in \Omega$. Hence, we need $\gamma_t \to 0$ as $t \to \infty$.

3. For the SA algorithm, the iterate $x_t = x_t(\xi_{[t-1]})$ is a function of the i.i.d. historic sample $\xi_{[t-1]} = (\xi_1, \ldots, \xi_{t-1})$ of the generated random process, so x_t and $f(x_t)$ are random variables. We cannot use the previous error functions to measure the optimality, e.g. $[f(x_t) - f_*]$ and $||x_t - x_*||_2^2$. Instead, a more appropriate criterion would be consider the expectation or high probability results.

Theorem 21.3 [*Nemirovski, Juditsky, Lan, Shapiro, 2009*]
Assume $f(x)$ is μ-strongly convex, and $\exists M > 0$, s.t. $\mathbb{E}\left[||\nabla F(x,\xi)||_2^2\right] \leq M^2, \forall x \in \mathbf{X}$, then SA method with $\gamma_t = \gamma/t$ at iteration t where $\gamma > 1/2\mu$ satisfies the following two properties:

(1) $\mathbb{E}\left[||x_t - x_*||_2^2\right] \leq \frac{C(\gamma)}{t}$, *where* $C(\gamma) = \max\{\frac{\gamma^2 M^2}{2\mu\gamma - 1}, ||x_1 - x_*||_2^2\}$, *and*

(2) If $f(X)$ is L-smooth and $x_* \in int(\mathbf{X})$, then

$$\mathbb{E}\left[f(x_t) - f_*\right] \leq \frac{LC(t)}{2t}.$$

Proof:

(1) For any given x_t and $\xi_{[t-1]}$, we want to calculate x_{t+1} by a sample ξ_t generate in this iteration, and the distance of x_{t+1} to the optimal x_* is

$$\begin{aligned} ||x_{t+1} - x_*||_2^2 &= ||\Pi_{\mathbf{X}}(x_t - \gamma_t \nabla F(x_t, \xi_t)) - \Pi_{\mathbf{X}}(x_*)||_2^2 \quad \text{(by definition)} \\ &\leq ||x_t - \gamma_t \nabla F(x_t, \xi_t) - x_*||_2^2 \quad \text{(by the non-expensiveness of projection)} \\ &= ||x_t - x_*||_2^2 - 2\gamma_t \langle \nabla F(x_t, \xi_t), x_t - x_* \rangle + \gamma_t^2 ||\nabla F(x_t, \xi_t)||_2^2. \end{aligned}$$

Because $\xi_{[t-1]}$ are samples generated from a random process, and x_t is a function of $\xi_{[t-1]}$, we take expectation on both sides of the above inequality to get

$$\begin{aligned} \mathbb{E}\left[||x_{t+1} - x_*||_2^2\right] \leq \quad & \mathbb{E}\left[||x_t - x_*||_2^2\right] - 2\gamma_t \mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle\right] + \gamma_t^2 \mathbb{E}\left[|\nabla F(x_t, \xi_t)||_2^2\right] \\ = \quad & \mathbb{E}\left[||x_t - x_*||_2^2\right] - 2\gamma_t \mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle\right] + \gamma_t^2 M^2. \end{aligned} \tag{21.1}$$

We claim that $\mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle\right] = \mathbb{E}\left[\langle \nabla f(x_t), x_t - x_* \rangle\right]$, which is shown below. Because $x_t = x_t(\xi_{[t-1]})$ is independent of ξ_t, we have

$$\begin{aligned} \mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle\right] &= \mathbb{E}\left[\mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle | \xi_{[t-1]}\right]\right] \quad \text{(by tower property)} \\ &= \mathbb{E}\left[\langle \mathbb{E}\left[\nabla F(x_t, \xi_t) | \xi_{[t-1]}\right], x_t - x_* \rangle\right] \\ &= \mathbb{E}\left[\langle \nabla f(x_t), x_t - x_* \rangle\right] \quad \text{(by the independence of samples).} \end{aligned}$$

We also claim that $\mathbb{E}\left[\langle \nabla f(x_t), x_t - x_* \rangle\right] \geq \mu \mathbb{E}\left[||x_t - x_*||_2^2\right]$. By μ-strongly convexity of $f(x)$, we have

$$\begin{aligned} f(x) \text{ is } \mu\text{-strongly convex} \Leftrightarrow \quad & \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \mu ||x - y||_2^2 \quad \forall x, y \in \mathbf{X} \\ \Leftrightarrow \quad & \langle \nabla f(x), x - y \rangle \geq \mu ||x - y||_2^2 + \langle \nabla f(y), x - y \rangle \quad \forall x, y \in \mathbf{X} \end{aligned} \tag{21.2}$$

Note that by the optimality of x_*, we have $\langle \nabla f(x_*), x - x_* \rangle \geq 0$. Combining the optimality condition and Inequality (21.2), it follows that

$$\langle \nabla f(x_t), x_t - x_* \rangle \geq \mu ||x_t - x_*||_2^2 + \langle \nabla f(x_*), x_t - x_* \rangle \geq \mu ||x_t - x_*||_2^2 \quad \forall x_t \in \mathbf{X}.$$

Taking expectation of the inequality above, we get

$$\mathbb{E}\left[\langle \nabla F(x_t, \xi_t), x_t - x_* \rangle\right] = \mathbb{E}\left[\langle \nabla f(x_t), x_t - x_* \rangle\right] \geq \mu \mathbb{E}\left[||x_t - x_*||_2^2\right]. \tag{21.3}$$

Putting Inequality (21.3) back to Inequality (21.1), we get the following result

$$\mathbb{E}\left[||x_{t+1} - x_*||_2^2\right] \leq (1 - 2\mu\gamma_t)\mathbb{E}\left[||x_t - x_*||_2^2\right] + \gamma_t^2 M^2.$$

Remember that we choose $\gamma_t = \gamma/t$ for iteration t, and $\gamma \geq 1/2\mu$, the inequality above is equivalent with

$$\mathbb{E}\left[||x_{t+1} - x_*||_2^2\right] \leq (1 - \frac{2\mu\gamma}{t})\mathbb{E}\left[||x_t - x_*||_2^2\right] + \frac{\gamma^2 M^2}{t}.$$

By induction, we conclude that $\mathbb{E}\left[||x_t - x_*||_2^2\right] \leq \frac{C(\gamma)}{t}$, where $C(\gamma) = \max\{\frac{\gamma^2 M^2}{2\mu\gamma - 1}, ||x_1 - x_*||_2^2\}$.

(2) Give any x_t and $\xi_{[t-1]}$, it can be shown that $f(x_t) - f(x_*) \leq \nabla f(x_*)^\top (x_t - x_*) + \frac{L}{2}||x_t - x_*||_2^2$ since we assume $f(x)$ is L-smooth. Optimality condition gives us that $\nabla f(x_*)^\top (x_t - x_*) \geq 0 \ \forall x_t \in \mathbf{X}$, thus $f(x_t) - f(x_*) \leq \frac{L}{2}||x_t - x_*||_2^2$. Taking expectation on both sides and combining the result in (1), we get

$$\mathbb{E}\left[f(x_t) - f_*\right] = \mathbb{E}\left[f(x_t) - f(x_*)\right] \leq \frac{L}{2}\mathbb{E}\left[||x_t - x_*||_2^2\right] \leq \frac{LC(t)}{2t}.$$

■

Remark 3.

1. Note that from Theorem 21.3, in order to get ϵ-accuracy, we need $\mathcal{O}(\frac{1}{\epsilon})$ number of samples in SA method, while we need $\mathcal{O}(\frac{1}{\epsilon^2})$ number of samples if using SAA method.
2. In the deterministic case (which is shown in Lecture 9), for strongly convex objective function, the error is $||x_t - x_*||_2^2 \leq \mathcal{O}\big((\frac{L-\mu}{L+\mu})^{2t}\big)$ which gives linear convergence rate. However, in the stochastic case, the expected error is $\mathbb{E}\left[||x_t - x_*||_2^2\right] \leq \mathcal{O}(\frac{1}{t})$ which gives a sublinear convergence rate.

We will discuss more on the sample complexity of SA in the next lecture.

References

[KS00] Kleywegt, A. and Shapiro, A. (2000). *Chapter 101 Stochastic Optimization.* http://www2.isye.gatech.edu/~anton/stochoptiebook.

[KSH01] Kleywegt, A, Shapiro, A and Homen-de-mello, T (2001). *The Sample Average Approximation Method for Stochastic Discrete Optimization*, (Vol. 12, No. 2, pp. 479-502). SIAM J. OPTIM.

[RM51] Robbins, H and Monro, S (1951). *A Stochastic Approximation Method*, (pp. 400–407). The annals of mathematical statistics.

[NJLS09] Nemirovski, A., Juditsky, A., Lan, G. and Shapiro, A. (2009). *Robust Stochastic Approximation Approach to Stochastic Programming*, (Vol. 19, No. 4, pp. 1574-1609). SIAM J. OPTIM.

IE 598: Big Data Optimization **Fall 2016**

Lecture 22: Stochastic Optimization II – November 08

Lecturer: Niao He *Scriber: Jialin Song*

Overview: In the last lecture we have introduced two approaches i.e., sample average approximation (SAA) and stochastic approximation (SA) to solve the stochastic optimization problems. In this lecture, we continue our discussion on stochastic approximation (SA), which is also known as stochastic gradient descent. We also extend our scope to solving non-smooth and general convex stochastic problems with mirror descent stochastic approximation. Finally, we discuss briefly on how to further improve stochastic gradient descent.

22.1 Recap of SAA and SGD

The stochastic optimization problem can often be formulated as follows

$$\min_{x\in X} f(x) := \mathbb{E}_\xi[F(x,\xi)] \tag{P}$$

We have discussed two approaches to tackle stochastic optimization problem:

Sample Average Approximation (SAA) First, we obtain N i.i.d. sample $\xi_1, \xi_2, ..., \xi_n$ of random variable ξ from a distribution P; we then adopt some first order or second order method to solve the following deterministic optimization problem

$$\min_{x\in X} f^N(x) = \frac{1}{N}\sum_{i=1}^{N} F(x,\xi_i) \tag{SAA}$$

Suppose the method produces a candidate solution x_t when solving (SAA) after t iterations.

- The convergence performance can be measured as the following way:

$$f(x_t) - f(x_*) = \underbrace{f(x_t) - f(x_*^N)}_{\text{optimization error}} + \underbrace{f(x_*^N) - f(x_*)}_{\text{estimation error}}$$

  Note that the optimization error depends on the algorithms utilized to solve the deterministic problem and sample size N, yet the estimation error only depends on N.

- In the last lecture, we show that if the sample size is

$$N = \mathcal{O}\left(\frac{\max_{x\in X} \mathrm{Var}[F(x,\xi)]}{\epsilon^2}\log(\frac{|X|}{\alpha})\right)$$

  then we have $\mathbb{P}(f(x_*^N) \le f(x_*) + \epsilon) \ge 1-\alpha$. Note that this only accounts to the estimation error.

Stochastic Approximation (SA, or SGD) The algorithm directly solves the original problem (P):

$$x_{t+1} = \Pi_X(x_t - \gamma_t \nabla F(x_t, \xi_t))$$

where $\nabla F(x_t, \xi_t)$ is called stochastic gradient.

- When f is L-smooth and μ-strongly convex, $x_* \in \text{int(X)}$, $\gamma_t = \mathcal{O}(\frac{1}{\mu t})$, we have the following result on the expectation of $f(x_t) - f(x_*)$

$$\mathbb{E}[f(x_t) - f(x_*)] \leq \mathcal{O}\left(\frac{\max_{x \in X} \mathbb{E}[\|\nabla F(x, \xi)\|_2^2]}{\mu^2 t}\right)$$

  Markov inequality gives us

$$\mathbb{P}(f(x_t) - f(x_*) \geq \epsilon) \leq \frac{\mathbb{E}[f(x_t) - f(x_*)]}{\epsilon} \leq \mathcal{O}\left(\frac{\max_{x \in X} \mathbb{E}[\|\nabla F(x, \xi)\|_2^2]}{\mu^2 t \epsilon}\right)$$

  If we set the total number of iterations $T = \mathcal{O}(\frac{M^2}{\mu^2 \epsilon \alpha})$, we have $\mathbb{P}(f(x_t) \leq f(x_*) + \epsilon) \geq 1 - \alpha$.

Deterministic Optimization vs. Stochastic Optimization Recall that for deterministic optimization problem

$$\min_{x \in X} f(x)$$

where f admits exact gradient information, we showed that

- $\diamond$ f is smooth and μ-strongly convex $\Longrightarrow \mathcal{O}((\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^{2t})$ (Accelerated Gradient Descent)
- $\diamond$ f is smooth and convex $\Longrightarrow \mathcal{O}(\frac{1}{t^2})$ (Accelerated Gradient Descent)
- $\diamond$ f is non-smooth and convex $\Longrightarrow \mathcal{O}(\frac{1}{\sqrt{t}})$ (Subgradient/Mirror Descent)

We see that the complexity depends heavily on the smoothness of the convex function. Also, in the case of stronly convex and smooth problems, there is a huge discrepancy between the linear rate by GD and sublinear rate by SGD. In this lecture, we intend to address the following questions?

- Is SGD an optimal algorithm for solving stochastic optimization problem?
- Does smoothness make any difference of the convergence rate?
- What happens when f is some general convex function?
- How can we improve upon SGD?

22.2 Lower Bounds Results on Stochastic Gradient Descent

In this section, we discuss the lower bound complexity results for solving stochastic optimization problems.

A Simple Example. Let us consider the one-dimensional function $f(x) = \mathbb{E}[\frac{1}{2}(x - \xi)^2]$, where $\xi \sim N(0, 1)$ is a standard normal random variable. Based on stochastic gradient descent we have

$$x_{t+1} = x_t - \gamma_t(x_t - \xi_t)$$

Let $x_1 = 0, \gamma_t = \frac{1}{t}$ and by induction we have

$$x_{t+1} = \frac{1}{t}\sum_{t=1}^{t} \xi_\tau$$

Hence, $x_{t+1} \sim N(0, \frac{1}{t})$ is a normal random variable with mean 0 and variance $\frac{1}{t}$.

Since $f(x) = \frac{1}{2}(x^2 + 1)$, it can be easily found taht the optimal solution is $x_* = 0$. Therefore

$$\mathbb{E}[\|x_{t+1} - x_*\|_2^2] = \frac{1}{t}$$

This implies that the $O(1/t)$ rate achieved by SGD is indeed tight.

Recall that for deterministic convex optimization problems, we show that for problems with sufficiently large dimensions, we cannot improve the $O(1/t^2)$ rate for smooth convex problems. In fact, for stochastic optimization problem with strongly convex objectives, we cannot get rates better than $\mathcal{O}(1/t)$, for any arbitrary dimensions.

To achieve a more general characterization for nonsmooth stochastic optimization problems, we introduce the notion of stochastic oracle.

Stochastic Oracle: given an input x, stochastic oracle returns $G(x, \xi)$ s.t.,

$$\mathbb{E}[G(x, \xi)] \in \partial f(x) \text{ and } \mathbb{E}[\|G(x, \xi)\|_p^2] \leq M^2$$

for some positive constant M and some $p \in [1, \infty]$. This characterizes the first and second moment of the estimator of true subgradient.

It was shown in [Nemirovsky and Yudin, 1983] that in the worst case,

- ◇ For convex problems, total number of stochastic oracles required is at least $T = \mathcal{O}(\frac{1}{\epsilon^2})$
- ◇ For strongly convex problems, total number of stochastic oracles required is at least $T = \mathcal{O}(\frac{1}{\epsilon})$

Theorem 22.1 ***[Agarwal et al., 2012]*** *Let $X = B_\infty(r)$ be a ℓ_∞ ball with radius bounded by r.*

(1) $\exists\ c_0 > 0$, $\exists$ a convex function f on R^d, $|f(x) - f(y)| < M\|x - y\|_\infty$, then for any algorithm making T stochastic oracles with $1 \leq p \leq 2$ and generating a solution x_T,

$$\mathbb{E}[f(x_T) - f(x_*)] \geq \min\left\{c_0 Mr\sqrt{\frac{d}{T}},\ \frac{Mr}{144}\right\}$$

(2) $\exists\ c_1,\ c_2 > 0$,$\exists$ a μ-strongly convex function, for any algorithm making T stochastic oracles with $p = 1$ and generating a solution x_T,

$$\mathbb{E}[f(x_T) - f(x_*)] \geq \min\left\{c_1\frac{M^2}{\mu^2 T},\ c_2,\ Mr\sqrt{\frac{d}{T}},\ \frac{M^2}{1152\mu^2 d},\ \frac{Mr}{144}\right\}$$

Moreover, such lower bounds can be attained by the Mirror Descent Stochastic Approximation algorithm, which we discuss in details below.

22.3 Stochastic Mirror Descent

Analogous to the deterministic optimization scenario, **Mirror Descent Stochastic Approximation (a.k.a. Stochastic Mirror Descent)** is adopted to solve non-smooth problems.

Let $\omega(x)$ be a continuously differentiable and 1-strongly convex function w.r.t. some norm $\|\cdot\|$. A simple example of a distance-generating function is $\omega(x) = \frac{1}{2}\|x\|_2^2$. Define function $V(x,y) = \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$, which is called the Bregman distance.

The **mirror descent stochastic approximation** works as follows:

$$x_{t+1} = \arg\min_{x\in X}\{V(x,x_t) + \langle \gamma_t G(x_t,\xi_t),\ x\rangle\}$$

Theorem 22.3.1 [Nemirovski et al., 2009]

Let f be a convex function, $\Omega = \max_{x\in X} V(x,x_1)$. Let the candidate solution $\hat{x}_T =$ be the weighted average

$$\hat{x}_T = \frac{\sum_{t=1}^T \gamma_t x_t}{\sum_{t=1}^T \gamma_t}$$

(1) If there exists $M > 0$, s.t., $\mathbb{E}[\|G(x,\xi)\|_*^2] \le M^2$, $\forall x \in X$, then

$$\mathbb{E}[f(\hat{x}_T) - f(x_*)] \le \frac{\Omega + \frac{M^2}{2}\sum_{t=1}^T \gamma_t^2}{\sum_{t=1}^T \gamma_t}$$

(2) If there exists $M > 0$, s.t., $\mathbb{E}\left[\exp\left\{\|G(x,\xi)\|_*^2/M^2\right\}\right] \le \exp\{1\}$, then

$$\mathbb{P}\left\{f(\hat{x}_T) - f(x_*) \ge \frac{\sqrt{2}M\sqrt{\Omega}(12+2\Omega)}{\sqrt{T}})\right\} \le 2\exp\{-\Omega\}$$

Remark. Note that the condition in (2) essentially states that the stochastic subgradient has a light-tailed distribution. And if $T \ge \mathcal{O}\left(\frac{M^2\Omega}{\epsilon^2}\log^2(\frac{1}{\alpha})\right)$, we have $\mathbb{P}(f(\hat{x}_T) \le f(x_*) + \epsilon) \ge 1-\alpha$. We provide the simple proof for part (1) below.

Proof: Based on the optimality condition of the mirror descent stochastic approximation, we can have

$$\gamma_t(x_t - x_*)^T G(x_t,\xi_t) \le V(x_t,x_*) - V(x_{t+1},x_*) + \frac{\gamma_t}{2}\|G(x_t,\xi_t)\|_*^2 \tag{22.1}$$

Rewrite (22.1) as follows

$$\gamma_t(x_t - x_*)^T g(x_t) \le V(x_t,x_*) - V(x_{t+1},x_*) - \gamma_t(G(x_t,\xi_t) - g(x_t))^T(x_t - x_*) + \frac{\gamma_t}{2}\|G(x_t,\xi_t)\|_*^2 \tag{22.2}$$

where $g(x_t) \in \partial f(x_t)$. Taking summation over $t = 1,...,T$, we have

$$\sum_{t=1}^T \gamma_t(x_t - x_*)^T g(x_t) \le V(x_1,x_*) + \sum_{t=1}^T \frac{\gamma_t^2}{2}\|G(x_t,\xi_t)\|_*^2 - \sum_{t=1}^T \gamma_t(G(x_t,\xi_t) - g(x_t))^T(x_t - x_*) \tag{22.3}$$

Let's set $\hat{x}_T = \frac{\sum_{t=1}^T \gamma_t x_t}{\sum_{t=1}^T \gamma_t}$, and consider the convexity of $f(x)$, we have

$$\sum_{t=1}^T \gamma_t(x_t - x_*)^T g(x_t) \ge \sum_{t=1}^T \gamma_t(f(x_t) - f(x_*)) \ge \left(\sum_{t=1}^T \gamma_t\right)(f(\hat{x}_T) - f(x_*)) \tag{22.4}$$

Combine (22.3) and (22.4), we can get

$$f(\hat{x}_T) - f(x_*) \leq \frac{V(x_1, x_*) + \sum_{t=1}^T \frac{\gamma_t^2}{2} \|G(x_t, \xi_t)\|_*^2 - \sum_{t=1}^T \gamma_t (G(x_t, \xi_t) - g(x_t))^T (x_t - x_*)}{\sum_{t=1}^T \gamma_t} \tag{22.5}$$

Taking expectations on both sides of (22.5), we can have

$$\mathbb{E}[f(\hat{x}_T) - f(x_*)] \leq \frac{\max_{x \in X} V(x, x_1) + \frac{M^2}{2} \sum_{t=1}^T \gamma_t^2}{\sum_{t=1}^T \gamma_t}$$

as desired. ■

22.4 Improving Stochastic Gradient Descent

Although we cannot improve the convergence rate the Stochastic Gradient Descent (SGD) or Stochastic Mirror Descent (SMD) methods, we can still try to accelerate the performance by improving the constant factors. We discuss several strategies below.

- ◇ **Reduce Variance:**
  - **Mini Batch sampling:** use a small batch of samples instead of one to estimate the gradient at every iteration

    $$G(x_t, \xi_t) \Longrightarrow \frac{1}{b} \sum_{i=1}^b G(x_t, \xi_{t,i})$$

    Consequently, the variance of the new stochastic gradient will be $O(b)$ times smaller, i.e. the constant term M^2 in the convergence now reduces to M^2/b.
  - **Importance Sampling:** Instead of sampling from $\xi \sim P$, we can obtain samples from another well defined random variable η with nominal distribution Q, and use a different stochastic gradient,

    $$G(x_t, \xi_t) \Longrightarrow G(x_t, \eta_t) \frac{P(\eta_t)}{Q(\eta_t)}$$

    The variance of the new stochastic gradient under properly chosen distribution Q could be smaller.
- ◇ **Adaptive Stepsize** The traditional fixed stepsize $\gamma_t = \frac{1}{\mu t}$ may be too small so that the efficiency of the stochastic gradient descent approach can be compromised. One may instead select the stepsize adaptively to optimize the progress at each iteration. For instance, in [YNS12], the authors propose to automatically update the stepsize based on the recursion

  $$\gamma_t = \frac{1}{\mu t} \Longrightarrow \gamma_t = \gamma_{t-1}(1 - c\gamma_{t-1})$$

- ◇ **Adaptation of Bregman Distance** One may also adaptively choose the Bregman distance and hope to improve the efficiency. For instance, the AdaGrad algorithm in [DHS11] propose the following

  $$\omega(x) = \frac{1}{2} x^T x \Longrightarrow \omega_t(x) = \frac{1}{2} x^T H_t x, \text{ where } H_t = \delta \mathbf{I} + [\sum_{t=1}^t g_t g_t^T]^{\frac{1}{2}}$$

  where $g_t = G(x_t, \xi_t)$.

References

[NY83] A. Nemirovski and D. Yudin, Problem Complexity and Method Efficiency in Optimization *Wiley, New York, 1983*

[NJLS09] A. Nemirovski, A. Juditsky, G. Lan and A. Shapiro, Robust Stochastic Approximation Approach to Stochastic Programming, *SIAM J. Optim. 19(4) (2009), pp. 1574?1609*

[YNS12] F. Yousefian, A. Nedich and U.V. Shanbhag, On Stochastic Gradient and Subgradient Methods with Adaptive Steplength sequences, *Automatica 48 (1) 56-67, 2012*

[ABRW12] A. Agarwal, P. Bartlett, P. Ravikumar and M. Wainwright, Information-theoretic Lower Bounds on the Oracle Complexity of Convex Optimization *IEEE Transactions on Information Theory, 58(5), 2012*

[DHS12] J. Duchi, E. Hazan, and Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research, 12(Jul), 2121-2159, 2012*

IE 598: Big Data Optimization **Fall 2016**

Lecture 23: Incremental Gradient Methods – November 10

Lecturer: Niao He *Scribers: Shripad Gade*

Overview: In this lecture we study incremental gradient algorithms for finite sum optimization problems. Incremental gradient algorithms enjoy both the linear convergence (like gradient descent) and the cheap iteration cost (like stochastic gradient descent). We will provide a brief survey on the recently developed incremental gradient algorithms, including SAG, SAGA, SVRG, S2GD, Finito, etc. Specifically, we will analyze the convergence of the Stochastic Variance Reduced Gradient (SVRG) algorithm.

23.1 Finite Sum Problems

Problems where the objective function can be defined as a finite sum of functions, are called finite sum problems, or big-n problem. Formally, a finite sum problem can be written as,

$$\min_{x\in\mathbb{R}^d} \quad f(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{23.1}$$

Notice that the structure is similar to sample average approximation described in Lecture 21. The number of functions, n, is analogous to the sample size drawn in Monte Carlo sampling (i.i.d. samples). Such type of problems are popular in many applications.

- **Empirical risk minimizatin:** In machine learning problems, the risk associated with a hypothesis (h) is approximated by an empirical risk $R(h)$, defined as the loss over the dataset $(x_1, y_1), \ldots, (x_n, y_n)$. Empirical risk given by $R(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y_i))$, has the structure of a finite sum problem.
- **Distributed optimization:** Distributed optimization [Ned09] involves a finite sum problem being solved by a group of computational entities (agents). Each agent has access to only a part of the finite sum, however by using an iterative consensus and local gradient based algorithm, one can show the convergence of local state estimate to the optimum. Interested readers are pointed to Prof. Tsitsiklis PhD thesis [Tsi84] which is one of the first works that discussed consensus and distributed optimization ideas.

We are interested in solving convex finite sum optimization problem as posed in (23.1). Suppose $f(x)$ is L-smooth and μ-strongly convex. We have studied two methods for solving convex optimization problem: Gradient Descent method, and Stochastic Gradient Descent. We begin by summarizing convergence results for both these algorithms when applied to the finite sum optimization problem.

Gradient Descent

The gradient descent update equation can be written as,

$$x^{t+1} = x^t - \eta\nabla f(x^t) = x^t - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x^t). \tag{23.2}$$

We can summarize the convergence properties derived in previous lectures,

- Convergence rate: linear rate, $\mathcal{O}((\frac{\kappa-1}{\kappa+1})^{2t})$, where $\kappa = \frac{L}{\mu}$ is the condition number.
- Iteration cost: $\mathcal{O}(n)$ ... (n gradients are computed for each iteration).
- Overall complexity: $\mathcal{O}(n\,\frac{L}{\mu}\log(\frac{1}{\epsilon}))$.

Stochastic Gradient Descent

The finite sum problem can be rewritten as a stochastic optimization problem,

$$\min_{x\in\mathbb{R}^d} \; f(x) = \mathbb{E}_I[f_I(x)], \tag{23.3}$$

where I is a uniform discrete random variable so that $\mathbb{P}(I=i)=\frac{1}{n}$. The SGD update equation is given by,

$$x^{t+1} = x^t - \eta^t \nabla f_{i_t}(x^t), \text{ where } i_t \sim I \tag{23.4}$$

We can summarize the convergence properties of SGD as derived in previous lectures,

- Convergence rate: sub-linear rate, $\mathcal{O}(\frac{L}{\mu^2 t})$.
- Iteration cost: $\mathcal{O}(1)$ ... (Only 1 gradient is computed for each iteration).
- Overall complexity: $\mathcal{O}(\frac{L}{\mu^2\epsilon})$.

Motivation

To summarize, when solving problems with finite sum structure

- GD enjoys a linear rate but $O(n)$ iteration cost
- SGD enjoys a sublinear rate but with $O(1)$ iteration cost

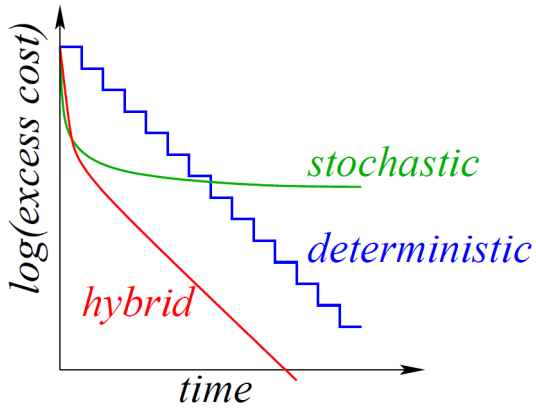


Figure 23.1: GD vs. SGD (Fig from [Bach13])

We would like to develop algorithms that have linear convergence (like GD) and cheap iteration cost (like SGD).

23.2 Brief Survey on Incremental Gradient Algorithms

Incremental gradient descent algorithms were developed to have such characteristics, and hence form an important class of algorithms. A list of few popular incremental algorithms is given below. A detailed summary and comparison among these algorithms is provided in Table 23.1.

- Deterministic Incremental Gradient Algorithms
  - *Incremental Gradient Descent* (IGD) - Bertsekas, 1997 [Ber97]
  - *Incremental Aggregated Gradient* (IAG) - Blatt *et.al*, 2007 [Bla07]
- Stochastic Incremental Gradient Algorithms
  - *Stochastic Average Gradient* (SAG) - Schmidt *et al.*, 2013 [Sch13]
  - *SAGA* - Defazio *et al.*, 2014 [Def14]
  - *Stochastic Variance Reduced Gradient* (SVRG) - Johnson *et al.*, 2013 [Joh13]
  - *Semi-Stochastic Gradient Descent* (S2GD) - Konecny *et al.*, 2014 [Kon13]
  - *FINITO* - Defazio *et al.*, 2014 [Def14b]
  - *Miminization by Incremental Surrogate Optimization* (MISO) - Mairal, 2013 [Mai13]
  - *Randomized Primal-Dual Gradient* [RPDG] - Lan *et al.*, 2015 [Lan15]

23.3 Variance Reduction Techniques

Suppose we want to estimate $\Theta = \mathbb{E}[X]$, the expected value of a random variable X. Suppose we also have access to a random variable Y which is highly correlated with X, and we can compute $\mathbb{E}[Y]$ easily. Let's consider the following point estimator $\hat{\Theta}_\alpha$ with $\alpha \in [0, 1]$:

$$\hat{\Theta}_\alpha = \alpha(X - Y) + \mathbb{E}[Y] \tag{23.5}$$

The expectation and variance are given by,

$$\mathbb{E}[\hat{\Theta}_\alpha] = \alpha\mathbb{E}[X] + (1 - \alpha)\mathbb{E}[Y] \tag{23.6}$$

$$\mathrm{Var}[\hat{\Theta}_\alpha] = \alpha^2 \left(\mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}[X, Y]\right) \tag{23.7}$$

Note that

- $\alpha = 1$, this estimator becomes $(X - Y) + \mathbb{E}[Y]$, which is an unbiased estimator.
- $\alpha = 0$, this estimator reduces to a constant $\mathbb{E}[Y]$, which has zero variance but could be heavily biased.
- If $\mathrm{Cov}[X, Y]$ is sufficiently large, then $\mathrm{Var}[\hat{\Theta}_\alpha] < \mathrm{Var}[X]$. The new estimator $\hat{\Theta}_\alpha$ has smaller variance than the direct estimator X.
- As α increases from 0 to 1, the bias decreases and the variance increases.

Recently developed incremental gradient algorithms namely SAG, SAGA, SVRG and S2GD are all special cases of the general variance reduction technique described above. See explanation below Table 23.1.

Algorithm	Key Step	Convergence Result	Notes
SAG	$x^{t+1} = x^t - \gamma \frac{1}{n} \sum_{i=1}^n g_i^t \qquad g_i^t = \begin{cases} \nabla f_{i_t}(x^t) \\ g_i^{t-1} \end{cases}$	$\gamma = \frac{1}{16L}$, Convergence: $\mathcal{O}\left(\left[1 - \min\{\frac{\mu}{16L}, \frac{1}{8n}\}\right]^t\right)$	High Memory Cost: $\mathcal{O}(n)$
SAGA	$x^{t+1} = x^t - \gamma \left[\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n} \sum_{i=1}^n g_i^{t-1}\right]$	$\gamma = \frac{1}{3L}$, Convergence: $\mathcal{O}\left(\left[1 - \min\{\frac{\mu}{3L}, \frac{1}{4n}\}\right]^t\right)$	High Memory Cost: $\mathcal{O}(n)$
SVRG	At epoch "s", do "m" steps: $x^{t+1} = x^t - \gamma \left[\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\tilde{x}) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x})\right]$ where, $\tilde{x}$ is the last iterate of epoch $s-1$	$\gamma < \frac{1}{2L}$, Convergence: $\mathcal{O}\left(\left[\frac{1}{\mu\gamma(1-2L\gamma)m} + \frac{2L\gamma}{1-2L\gamma}\right]^s\right)$	Low Memory Cost Require two loops
S2GD	Same as above, but with random number of inner step See [Kon13] for algorithm.	$\gamma < \frac{1}{2L}$, $\nu \le \mu$, $\beta \triangleq \sum_{t=1}^m (1-\nu\gamma)^{m-t}$ Convergence: $\mathcal{O}\left(\left[\frac{(1-\nu\gamma)^m}{\mu\gamma\beta(1-2L\gamma)} + \frac{2(L-\mu)\gamma}{1-2L\gamma}\right]^s\right)$	Low Memory Cost Require two loops
FINITO	$x^{t+1} = \frac{1}{n} \sum_{i=1}^n \phi_i^t - \frac{1}{\alpha\mu n} \sum_{i=1}^n \nabla f_i(\phi_i^t)$	$n \gg \frac{L}{\mu}$, Convergence: $\mathcal{O}\left(\left[1 - \frac{1}{2n}\right]^t\right)$	Memory cost higher than either SAG or SAGA. State $\phi \to \mathcal{O}(n)$ Gradient $\nabla f_i \to \mathcal{O}(n)$
MISO	See [Mai13] for algorithm.	$\mathcal{O}\left(\left[(1-\delta) + \delta\frac{L}{\rho+u}\right]^{t-1}\right)$	Memory cost higher than either SAG or SAGA. State $\phi \to \mathcal{O}(n)$ Gradient $\nabla f_i \to \mathcal{O}(n)$

Table 23.1: Summary of Incremental Gradient Algorithms and their Convergence properties.

Connection to variance reduction technique: Let $\Theta = \nabla f(x_t)$, $X = \nabla f_{i_t}(x_t)$, the gradient used in the above algorithms can be conceived as special cases $\hat{\Theta}_\alpha$ defined in (23.5):

- SGD: $x^{t+1} = x^t - \gamma^t \nabla f_{i_t}(x^t)$ $(\alpha = 1, Y = 0)$
- SAG: $x^{t+1} = x^t - \gamma \left[\frac{1}{n}(\nabla f_{i_t}(x^t) - g_{i_t}^{t-1}) + \frac{1}{n} \sum_{i=1}^n g_i^{t-1}\right]$, $(\alpha = \frac{1}{n}, Y = g_{i_t}^{t-1})$
- SAGA: $x^{t+1} = x^t - \gamma \left[\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n} \sum_{i=1}^n g_i^{t-1}\right]$, $(\alpha = 1, Y = g_{i_t}^{t-1})$
- SVRG, S2GD: $x^{t+1} = x^t - \gamma \left[\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\tilde{x}) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x})\right]$, $(\alpha = 1, Y = \nabla f_{i_t}(\tilde{x}))$

23.4 Stochastic Variance Reduced Gradient Algorithm

Stochastic Gradient Descent (SGD) is popular among large scale optimization practitioners, however, it has a slower rate of convergence due to inherent variance [Joh13]. Consider a finite-sum optimization problem given in (23.1). The gradient descent update is given by Eq. 23.2,

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

However, since this method requires n gradient computations, Stochastic Gradient Descent algorithm has replaced it as a popular solution. The stochastic gradient descent update is given by Eq. 23.4,

$$x^{t+1} = x^t - \eta^t \nabla f_{i_t}(x^t). \qquad \dots (\mathbb{P}(i_t = i) = \frac{1}{n})$$

Notice that in expectation both gradient descent and stochastic gradient descent are identical, i.e. $\mathbb{E}[x^t|x^{t-1}] = x^{t-1} - \eta^t \left(\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^{t-1})\right)$ for both GD as well as SGD. However, SGD has a disadvantage in the sense that the gradient used in every iteration equals $\nabla f(x)$ only in expectation and may have large deviation from the mean in some of the instances. Due to inherent large variance, we observe only a sublinear $\mathcal{O}(1/t)$ rate of convergence. Incremental algorithms have become popular in the past decade due to their linear rate of convergence (improved from sublinear rate, see Section 23.2 and Table 23.1 for list of methods and their convergence rates).

We will study one of the popular incremental algorithms called Stochastic Variance Reduced Gradient (SVRG). The strengths of this work are [Joh13],

- SVRG algorithm does not require storage of gradients as seen in SAG or SAGA.
- Convergence rates for SVRG can be proved easily and a very intuitive explanation can be provided by linking increased speed to reduced variance.

Algorithm 1 Stochastic Variance Reduced Gradient

Parameters update frequency m and learning rate η
Initialize $\tilde{x}_0$
for $s = 1, 2, \dots$ **do**
  $\tilde{x} = \tilde{x}^{s-1}$
  $\tilde{\theta} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{x})$
  $x_0 = \tilde{x}$
  for $t = 1, 2, \dots, m$ **do**
    Randomly pick $i_t \in \{1, 2, \dots, n\}$ and update weight,
      $x^t = x^{t-1} - \eta \left(\nabla f_{i_t}(x^{t-1}) - \nabla f_{i_t}(\tilde{x}) + \tilde{\theta}\right)$
  end for
  Update $\tilde{x}^s$
    Option I $\tilde{x}^s = x^m$
    Option II $\tilde{x}^s = \frac{1}{m} \sum_{t=1}^{m} x^t$
    Option III $\tilde{x}^s = x^t$ for randomly chosen $t \in \{1, 2, \dots, m\}$
end for

Convergence Analysis

For simplicity we consider Problem 23.1, where $f_i(x)$ is convex and L-smooth for all $i = 1, \ldots, n$, and $f(x)$ is μ-strongly convex. For instance, if each of the function f_i is L_i-smooth, then once can set $L = \max_{i=1,\ldots,n} L_i$. We show linear convergence of SVRG for such a finite-sum optimization problem.

Theorem 23.1 *Assume $f_i(x)$ is is convex and L-smooth and $f(x)$ is μ-strongly convex. Let $x_* = \arg\min f(x)$. Assume m is sufficiently large (and $\eta < \frac{1}{2L}$), so that,*

$$\rho = \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta} < 1,$$

then we have geometric convergence in expectation for SVRG (under Option II and Option III), i.e.,

$$\mathbb{E}[f(\tilde{x}^s) - f_*] \leq \rho^s \, [f(\tilde{x}^0) - f_*].$$

Remark. Setting $\eta = \theta/L$ with some constant $\theta > 0$, this gives

$$\rho = \frac{L}{\mu\theta(1-2\theta)m} + \frac{2\theta}{1-2\theta} = O\left(\frac{L}{\mu m} + \text{ const.}\right)$$

Hence, if we set $m = O(L/\mu)$, then this will result in a constant rate ρ. The number of epochs needed to achieve an ϵ optimal solution is $O(\log(\frac{1}{\epsilon}))$. Therefore, the overall complexity for SVRG is

$$\mathcal{O}\left((m+n)\log\left(\frac{1}{\epsilon}\right)\right) = \mathcal{O}\left((n+\frac{L}{\mu})\log\left(\frac{1}{\epsilon}\right)\right)$$

Note that the complexity is significantly better that of Gradient Descent, i.e.

$$\mathcal{O}\left(n \cdot \frac{L}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$$

when the condition number L/μ is large.

Extensions.

1. **Non-uniform sampling:** SVRG algorithm assumes uniform sampling, however, one may choose an adaptive sampling rate,
$$\mathbb{P}(i_t = i) = \frac{L_i}{\sum L_i}$$
where L_i is the smoothness parameter for f_i. This sampling strategy improves the complexity from $\mathcal{O}\left(\left(n + \frac{L_{max}}{\mu}\right)\log(\frac{1}{\epsilon})\right)$ to $\mathcal{O}\left(\left(n + \frac{L_{avg}}{\mu}\right)\log(\frac{1}{\epsilon})\right)$. Intuitively, the function $f_i(x)$ that has a higher Lipschitz constant (which is prone to change relatively rapidly) gets higher probability of getting selected.

2. **Composite convex minimization:** These are problems of the form
$$\min_x \; \frac{1}{n}\sum_i f_i(x) + g(x)$$
where $f_i(x)$ are smooth and convex, but $g(x)$ is convex but possibly nonsmooth. Such problems can be handle by **prox-SVRG** [Xia14] by imposing an additional proximal operator of g at iteration.

3. **Acceleration:** We can accelerate SVRG further to arrive at an optimal complexity of

$$\mathcal{O}\left(\left(n+\sqrt{\frac{nL}{\mu}}\right)\log(\frac{1}{\epsilon})\right).$$

This improvement is significant in problems where $\frac{L}{\mu} \gg n$.

We now prove the main theorem. We start by proving provide the following lemma.

Lemma 23.2 *For any x, we have*

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x)-\nabla f_i(x_*)\|_2^2 \le 2L(f(x)-f(x_*)). \tag{23.8}$$

Proof: For any i, consider a function $g_i(x)$,

$$g_i(x) = f_i(x) - f_i(x_*) - \nabla f_i(x_*)^T(x-x_*).$$

Note that $\nabla g_i(x) = \nabla f_i(x) - \nabla f_i(x_*)$. Clearly, $\nabla g_i(x_*) = 0$, implying that $g_i(x_*) = \min_x g_i(x)$. Therefore, following the definition of minimum and the L-smoothness of function $g_i(x)$, we arrive at

$$0 = g_i(x_*) \le \min_{\eta}\left[g_i(x-\eta\nabla g_i(x))\right] \le g_i(x) - \frac{1}{2L}\|\nabla g_i(x)\|_2^2$$

That is,

$$\begin{aligned}\|\nabla g_i(x)\|_2^2 &\le 2Lg_i(x)\\ \|\nabla f_i(x)-\nabla f_i(x_*)\|_2^2 &\le 2L\left(f_i(x)-f_i(x_*)-\nabla f_i(x_*)^T(x-x_*)\right).\end{aligned}$$

By summing the above inequality for all $i = 1, 2, \ldots, n$ and using the fact $\nabla f(x_*) = 0$ and $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$, we have

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x)-\nabla f_i(x_*)\|_2^2 \le 2L(f(x)-f(x_*)).$$

■

We now proceed to prove the theorem.

Proof of Theorem. Let $v^t = \nabla f_{i_t}(x^{t-1}) - \nabla f_{i_t}(\tilde{x}) + \nabla f(\tilde{x})$. We now take expectation with respect to i_t conditioned on w^{t-1} and obtain,

$$\begin{aligned}
&\mathbb{E}[\|v^t\|^2] = \mathbb{E}\left[\|\left[\nabla f_{i_t}(x^{t-1}) - \nabla f_{i_t}(x_*)\right] + \left[\nabla f_{i_t}(x_*) - \nabla f_{i_t}(\tilde{x}) + \nabla f(\tilde{x})\right]\|_2^2\right] && \\
&\le 2\mathbb{E}\left[\nabla\|f_{i_t}(x^{t-1}) - \nabla f_{i_t}(x_*)\|_2^2\right] + 2\mathbb{E}\left[\|\nabla f_{i_t}(\tilde{x}) - \nabla f_{i_t}(x_*) - \nabla f(\tilde{x})\|_2^2\right] && (\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2)\\
&\le 2\mathbb{E}\left[\nabla\|f_{i_t}(x^{t-1}) - \nabla f_{i_t}(x_*)\|_2^2\right] + 2\mathbb{E}\left[\|\nabla f_{i_t}(\tilde{x}) - \nabla f_{i_t}(x_*)\|_2^2\right] && (\nabla f(\tilde{x}) = \mathbb{E}\left[\nabla f_{i_t}(x) - \nabla f_{i_t}(x_*)\right])\\
& && (\mathbb{E}[\|e-\mathbb{E}[e]\|_2^2] \le \mathbb{E}[\|e\|_2^2])\\
&\le 4L\left[f(x^{t-1}) - f(x_*) + f(\tilde{x}) - f(x_*)\right] && (\text{From Eq. 23.8})
\end{aligned}$$

Now notice from the definition of v^t, $\mathbb{E}\left[v^t|x^{t-1}\right] = \nabla f(x^{t-1})$; and this leads to,

$$\begin{aligned}
&\mathbb{E}\left[\|x^t - x_*\|_2^2\right] = \mathbb{E}\left[\|x^{t-1} - \eta v^t - x_*\|_2^2\right] \\
&= \|x^{t-1} - x_*\|_2^2 - 2\eta(x^{t-1} - x_*)^T\mathbb{E}\left[v^t\right] + \eta^2\mathbb{E}\left[\|v^t\|_2^2\right] \\
&\leq \|x^{t-1} - x_*\|_2^2 - 2\eta(x^{t-1} - x_*)^T\nabla f(x^{t-1}) + 4L\eta^2\left[f(x^{t-1}) - f(x_*) + f(\tilde{x}) - f(x_*)\right] \\
&\leq \|x^{t-1} - x_*\|_2^2 - 2\eta(f(x^{t-1}) - f(x_*)) + 4L\eta^2\left[f(x^{t-1}) - f(x_*) + f(\tilde{x}) - f(x_*)\right] \\
&\leq \|x^{t-1} - x_*\|_2^2 - 2\eta(1-2L\eta)(f(x^{t-1}) - f(x_*)) + 4L\eta^2\left[f(\tilde{x}) - f(x_*)\right]
\end{aligned}$$

We consider a fixed stage s, so that $\tilde{w} = \tilde{w}^{s-1}$ and $\tilde{w}^s$ is selected after all the updates have completed. By summing the previous inequality over $t = 1, 2, \ldots, m$, taking expectation with all the history, and using option II (or III) at stage s, we obtain [1],

$$\begin{aligned}
&\mathbb{E}\left[\|x^m - x_*\|^2\right] + 2\eta(1-2L\eta)m\mathbb{E}\left[f(\tilde{x}^s) - f(x_*)\right] \\
&\qquad \leq \mathbb{E}\left[\|x^0 - x_*\|^2\right] + 4Lm\eta^2\mathbb{E}\left[f(\tilde{x}^{s-1}) - f(x_*)\right] \\
&\qquad \leq \mathbb{E}\left[\|\tilde{x} - x_*\|^2\right] + 4Lm\eta^2\mathbb{E}\left[f(\tilde{x}^{s-1}) - f(x_*)\right] \\
&\qquad \leq \frac{2}{\mu}\mathbb{E}\left[f(\tilde{x}) - f(x_*)\right] + 4Lm\eta^2\mathbb{E}\left[f(\tilde{x}) - f(x_*)\right] \qquad (f(x) \text{ is } \mu \text{ strongly convex.})
\end{aligned}$$

Clearly, from the above inequality we get,

$$\mathbb{E}\left[f(\tilde{x}^s) - f(x_*)\right] \leq \left[\frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta}\right]\mathbb{E}\left[f(\tilde{x}^{s-1}) - f(x_*)\right].$$

This gives us the desired geometric convergence rate, $\mathbb{E}[f(\tilde{x}^s) - f_*] \leq \rho^s\,[f(\tilde{x}^0) - f_*]$. ■

References

[Ned09] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, 54.1 (2009): 48-61.

[Tsi84] J. Tsitsiklis, "Problems in Decentralized Decision making and Computation," Ph.D. Thesis, No. LIDS-TH-1424. MIT Cambridge, LIDS, 1984.

[Bach13] F. Bach, "stochastic gradient methods for machine learning," *Slide*, (2013).

[Ber97] D. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM Journal on Optimization* 7.4 (1997): 913-926.

[Bla07] D. Blatt and A. Hero and H. Gauchman "A convergent incremental gradient method with a constant step size," *SIAM Journal on Optimization* 18.1 (2007): 29-51.

[Sch13] M. Schmidt and N. Le Roux and F. Bach "Minimizing finite sums with the stochastic average gradient," *arXiv preprint*, arXiv:1309.2388 (2013).

[1] Note that here we use our choice of update equation (Option II or Option III). If we use Option II, we use the convexity of $f(x)$ to establish the following bound. If we use Option III, we need to use the fact that, since $\tilde{x}$ is chosen randomly from x^t, $\mathbb{E}\left[f(x^{t-1})\right] = \mathbb{E}\left[f(\tilde{x})\right]$.

$$-\sum_{t=1}^{m} 2\eta(1-2L\eta)\mathbb{E}\left[f(x^{t-1}) - f(x_*)\right] \leq -\sum_{t=1}^{m} 2\eta(1-2L\eta)m\mathbb{E}\left[f(\tilde{x}) - f(x_*)\right]$$

[Def14] A. Defazio and F. Bach and S. Lacoste-Julien "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in Neural Information Processing Systems* , 2014.

[Joh13] R. Johnson and T. Zhang "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in Neural Information Processing Systems*, 2013.

[Kon13] J. Konecny and P. Richtarik "Semi-stochastic gradient descent methods," *arXiv preprint*, arXiv:1312.1666 (2013)

[Def14b] A. Defazio and J. Domke and T. Caetano "Finito: A faster, permutable incremental gradient method for big data problems," *ICML*, 2014.

[Mai13] J. Mairal "Optimization with First-Order Surrogate Functions," *ICML*, 2013.

[Lan15] G. Lan and Y. Zhou "An optimal randomized incremental gradient method," *arXiv preprint* arXiv:1507.02000 (2015).

[Xia14] L. Xiao and T. Zhang "A proximal stochastic gradient method with progressive variance reduction," *arXiv preprint SIAM Journal on Optimization* 24(4), 2057-2075, 2014.

IE 598: Big Data Optimization **Fall 2016**

Lecture 24: Summary and Outlook – November 15

Lecturer: Niao He *Scribers: Joseph Lubars*

24.1 Case Study in Large-Scale Logistic Regression

24.1.1 Logistic Regression Model

In lecture 13, we looked at the performance of various algorithms on logistic regression. Given data $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in R^d, y_i \in \{-1, 1\}$, $i = 1, \ldots, N$, logistic regression arises from applying maximum likelihood estimation to the following likelihood function:

$$P(y = 1|x, w) = \sigma(w^T x) := \frac{1}{1 + \exp(-w^T x)}. \tag{24.1}$$

Applying a regularization term with λ as the regularization parameter, we get the following optimization problem:

$$\min_w f(w) := \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2} \|w\|_2^2 \tag{24.2}$$

Note that this is a smooth, strongly convex function. In practice, N will be very large, on the order of thousands or more. With such a large N, the first part of the above objective resembles an expectation, allowing us to use stochastic algorithms. Furthermore, given the structure of the problem as a finite sum of convex functions, we are in a position to use the incremental gradient methods from last lecture.

24.1.2 Algorithm Comparison

We will compare three algorithms for solving the optimization problem: Gradient Descent (GD), Stochastic Gradient Descent (SGD), and Stochastic Variance Reduced Gradient (SVRG). Their relative performance can be summed up in the following table:

Algorithm	Iteration Complexity
GD	$N \frac{L}{\mu} \log(\frac{1}{\epsilon})$
SGD	$\frac{L}{\mu^2} \cdot \frac{1}{\epsilon}$
SVRG	$(N + \frac{L}{\mu}) \log(\frac{1}{\epsilon})$

Notations:

1. μ is the strong convexity parameter
2. L is the constant for L-smoothness

24.1.3 Data Generation

We will do first do experiments using a synthetic dataset. In this dataset, we will use a relatively small N for large-scale problems, with $N = 10000$. We pick $\lambda = 10^{-4}$. This choice of an extremely small λ is intentional, so the regularization term does not dominate. In practice, a choice of $\lambda = 1/N$ is a good default choice and is what is used here. x_i is generated according to a standard normal random variable, normalized to norm 1. y_i is generated according to the logistic model (Equation 24.1). Because L scales as $\lambda + \|x_i\|_2^2/4$, we therefore set $L = \lambda + 1/4$.

Given these parameters, both N and L/μ are on the order of 10^4. Then, the coefficient of the $1/\epsilon$ term for complexity is on the order of 10^8 for both Gradient Descent and Stochastic Gradient Descent. However, for SVRG, $(N + L/\mu)$ is only on the order of 10^4. Because of this, we expect SVRG to have significantly better performance than the other two algorithms.

```
N = 10000
dim = 50
lamda = 1e-4
np.random.seed(1)
w = np.matrix(np.random.multivariate_normal([0.0]*dim, np.eye(dim))).T
X = np.matrix(np.random.multivariate_normal([0.0]*dim, np.eye(dim), size = N))
X = np.matrix(normalize(X, axis=1, norm='l2'))
y = 2 * (np.random.uniform(size = (N, 1)) < sigmoid(X*w)) - 1
```

Listing 1: Python Code for Data Generation

24.1.4 Optimization via CVXPY

First, we use CVXPY to obtain an optimal solution using the interior point method. We set the tolerance to 10^{-15} to allow us to compare our objective function value more accurately. As we will see, SVRG is able to give us similar accuracy in a fraction of the time.

```
import cvxpy as cvx
w = cvx.Variable(dim)
loss = 1.0/N * cvx.sum_entries(cvx.logistic(-cvx.mul_elemwise(y, X*w))) + lamda/2 * cvx.
    sum_squares(w)

problem = cvx.Problem(cvx.Minimize(loss))
problem.solve(verbose=True, abstol=1e-15)
opt = problem.value
print('Optimal Objective function value is: {}'.format(opt))
```

Listing 2: Python Code for CVXPY Optimization

In the end, we get an optimal objective value of 0.605366383115 after running for over 12 seconds.

24.1.5 Gradient Descent

We set the constant L as described above. We also fix the number of passes over the data for fair comparison between algorithms. Gradient descent gives us an objective value of 0.609, within 0.004 of the optimal value obtained from CVXPY above.

```
L = lambda + 1.0/4
num_pass = 50

## Define the objective and gradient oracles.
```

```
def obj(w):
    return 1.0/N * np.sum( np.log(1 + np.exp(-np.multiply(y, (X*w)))) ) + 1.0/2 * lamda * (w
    .T*w)

def grad(w,X,y):
    return 1.0/(X.shape[0]) * X.T * np.multiply( y, sigmoid(np.multiply(y, X*w)) - 1) +
    lamda*w

## Gradient Descent
w = np.matrix([0.0]*dim).T
obj_GD = []
max_iter = num_pass
for t in range(0, max_iter):
    obj_val = obj(w)
    w = w - 2.0/(L+lamda) * grad(w, X, y)
    obj_GD.append(obj_val.item())

print('Objective function value is: {}'.format(obj_GD[-1]))
```

Listing 3: Python Code for Gradient Descent

24.1.6 Stochastic Gradient Descent

Now, we use SGD. A batch size of 50 is used, improving the algorithm by averaging a number of data points together at each step. The total number of steps is adjusted so that the number of passes is the same as used for full gradient descent. Notably, the theoretical stepsize is not used here. Because $1/\lambda = 10000$ for this example, the theoretical stepsize would start out at 10000, which is much too large for convergence. This effect is mitigated in practice both by using $1/(t + t_0)$ to start with a smaller time-varying factor, and by adding a constant of 0.1 in front to further reduce the initial stepsize. A comparison of the performance with the two step sizes can be found in Figure 24.1. Because SGD is a stochastic algorithm, the objective does not decrease monotonically. The final objective value also varies between runs.

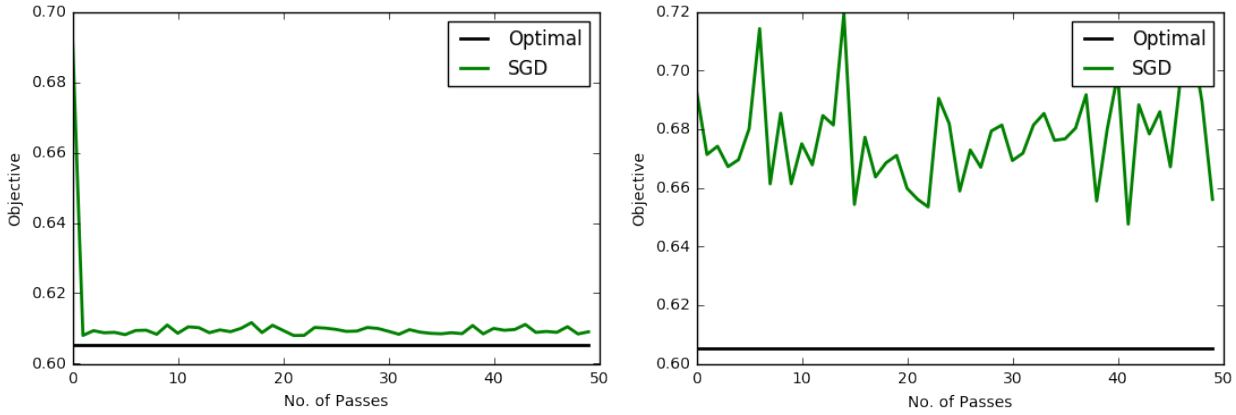
```
## Stochastic Gradient Descent
w = np.matrix([0.0]*dim).T
obj_SGD = []
batch = 50
for s in range(num_pass):
    obj_val = obj(w)
    obj_SGD.append(obj_val.item())
    max_iter = int(N/batch)
    for t in range(max_iter):
        rand_idx = np.random.randint(0, N-1,batch)
        yt = y[rand_idx, 0]
        xt = X[rand_idx, :]
        # gamma = 1/(lamda*(t+1))              # theoretical stepsize
        gamma = 0.1/(lamda*(t+100))            # better stepsize
        w = w -  gamma * grad(w,xt,yt)

print('Objective function value is: {}'.format(obj_SGD[-1]))
```

Listing 4: Python Code for Stochastic Gradient Descent

24.1.7 Stochastic Variance Reduced Gradient

In order to keep the number of passes through the data consistent, we set the number of epochs to 15. For each epoch, we pass through the data three times, so the total number of passes through the data is close to 50. we keep the minibatch size at 50 from SGD.



(a) SGD with better stepsize

(b) SGD with theoretical stepsize

Figure 24.1: A comparison of Stepsizes for SGD. With the theoretical stepsize, it never seems to converge.

```
w = np.matrix([0.0]*dim).T
obj_SVRG = []
passes_SVRG = []

Epochs = 15
k = 2
batch = 50
for s in range(Epochs):
    obj_val = obj(w)
    obj_SVRG.append(obj_val.item())
    passes_SVRG.append(s*k+s)

    w_prev = w
    gradient = grad(w, X, y)

    obj_SVRG.append(obj_val.item())
    passes_SVRG.append(s*k+s+1)

    max_iter = int(k*N/batch)
    for t in range(max_iter):
        rand_idx = np.random.randint(0, N-1, batch)
        yt = y[rand_idx, 0]
        xt = X[rand_idx, :]
        gamma = 1/L
        w = w - gamma * (grad(w,xt,yt) - grad(w_prev,xt,yt) + gradient)

print('Objective function value is: {}'.format(obj_SVRG[-1]))
```

Listing 5: Python Code for Stochastic Variance Reduced Gradient

The final objective value is even lower than the optimal value calculated from the interior point method, indicating that SVRG is converging to the optimal solution extremely quickly. In fact, this value is reached after just 6 epochs.

24.1.8 Comparing Algorithm Performance on Synthetic Data

Now that we have run all three algorithms (GD, SGD, SVRG) on our synthetic dataset, it is time to compare their performance. The relative objective error of the three algorithms can be found in Figure 24.2. Note that SVRG is tremendously better than the other two algorithms, achieving an error of 10^{-10} in just 12 passes through the data. The jumps in the graph are due to the fact that the solution is not updated with every pass through the data with SVRG. SGD is doing better than GD, but still does not approach the performance of SVRG.

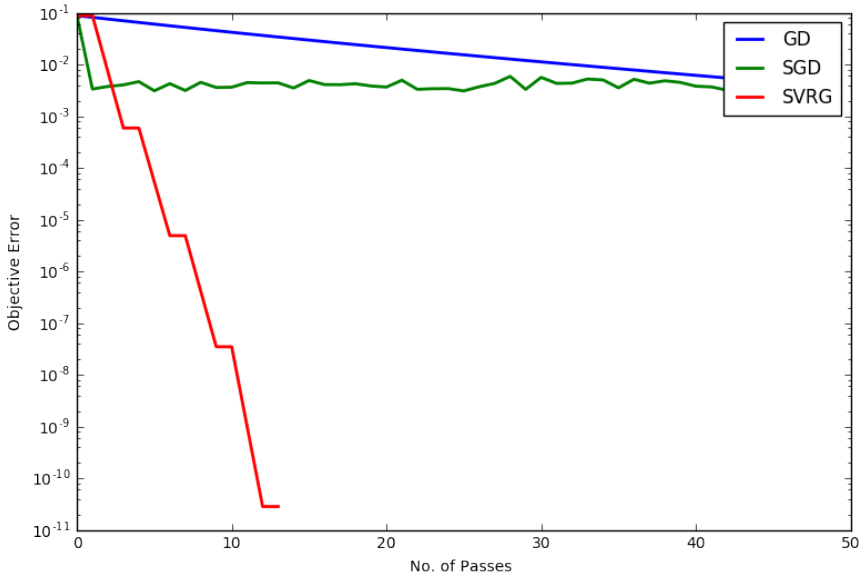


Figure 24.2: Comparison of GD, SGD, and SVRG on a synthetic dataset

24.1.9 Comparing Algorithm Performance on Real Data

Now, we use the Covertype dataset, where the forest cover type is predicted from cartographic information. This dataset is very popular in the machine learning community. In this test case, we are using $N = 58101$ and the dimension of the data is 54. The relative performance of the algorithms can be found in Figure 24.3. As you can see, SGD continues to significantly outperform normal gradient descent. SVRG beats the benchmark set by interior point method after just 12 passes through the data.

24.2 Course Summary and Outlook

In big data and machine learning, there are many different tasks, including:

1. Image/Signal Processing
2. Supervised Learning
3. Unsupervised Learning

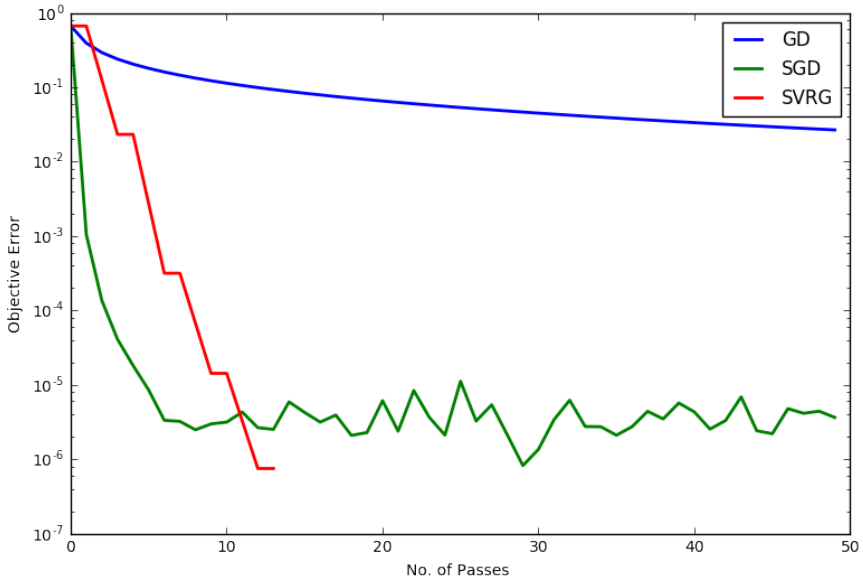


Figure 24.3: Comparison of GD, SGD, and SVRG on the Covertype dataset

4. Reinforcement Learning
5. Recommendation Systems

However, in the end, at the heart of all of these tasks is optimization. In this course, we have explored optimization theories, algorithms, and big data applications, gaining the tools to tackle these tasks. Along the way, we have learned how to use the structure of various problems to design optimization algorithms, and studied the complexity of algorithms required to solve these various classes of problems. With the tools from this class, we can identify which algorithm to use for any given problem.

What follows is a summary of all the topics that have been covered:

24.2.1 Summary of Topics Covered

The primary focus of this class has been convex optimization problems, because these are the problems that we can solve. Critically, any local optimum for a convex problem is also a global optimum, which is crucial in the operation of the algorithms that we have studied.

24.2.1.1 Convex Optimization Fundamentals

We learned the fundamentals of convex optimization, including definitions and characterizations of convex sets and convexity, operations that preserver convexity, and optimality conditions. We also learned about Lagrangian duality, saddle points, and KKT conditions. For more fundamentals, consider taking IE 521.

Conic optimization is a family that already covers most relevant convex problems. We learned three types of conic problems: LP, SOCP, and SDP. We also learned our first polynomial-time algorithm for solving a convex problem: the Interior Point Method.

24.2.1.2 Convex Optimization Algorithms

We learned a large number of algorithms and their iteration complexity, along with the situations where they are useful. In general, you should consider the structure of the problem and choose the algorithm that best utilizes this structure. For example, in the case study above, SVRG best utilizes the finite sum structure of the problem, and therefore outperforms SGD. Broken into classes, with a few notes, here is a rough list. See the lecture slides for the iteration complexities/costs:

- Smooth Convex Optimization
  - No restrictions: (Accelerated) Gradient Descent
  - Need to project: Projected Gradient Descent
  - Hard to calculate the projection: Frank-Wolfe
  - High-Dimensional: Block Coordinate Gradient Descent (BCGD)
- Nonsmooth Convex Optimization
  - Subgradient Descent
  - Mirror Descent (More general than Subgradient Descent)
  - When proximal operators are easy to compute: (Accelerated) Proximal Point Algorithm
- Nonsmooth Convex Optimization, in the form $f(x) = max_{y \in Y}\{\langle Ax + b, y\rangle - \phi(y)\}$
  - Often, problems can be represented as the maximum of convex functions, allowing you to use these algorithms:
  - Nesterov's Smoothing + (Accelerated) Gradient Descent
  - Mirror Prox: Good with an easy prox-mapping
- Nonsmooth Convex Optimization, in the form $\min_x f(x) + g(x)$ (e.g. LASSO)
  - (Accelerated) Proximal Gradient
  - Douglas-Rachford Splitting: Good when you can efficiently compute proximal operator for both f and g
  - Krasnosel'skii-Mann (KM): A generalization of fixed-point algorithms, splitting algorithms
- Stochastic Convex Optimization
  - Sample Average Approximation (SAA)
  - When f is strongly convex: Stochastic Approximation (SA)
  - For general convex functions: Mirror Descent SA
- Finite Sum of Convex Functions
  - When gradient is easy to calculate: Gradient Descent
  - For large N: Stochastic Gradient Descent
  - To utilize the sum structure: SVRG/S2GD

24.2.1.3 Topics Not Covered

We have covered a huge number of first-order algorithms, but there are still many topics that we did not cover. Within convex optimization, these topics include:

1. Nonsmooth First-Order Optimization Algorithms: There are still several algorithms that we did not cover, including bundle methods and primal-dual methods.
2. Stochastic Optimization Algorithms: The algorithms we have discussed have variations in stochastic situations (e.g. Stochastic Frank Wolfe)
3. Second-Order Methods: We have all learned Newton's Method, but there are other second-order methods with superlinear convergence rates.
4. Zero-Order Methods: Sometimes, even the gradient is too expensive to calculate. In these situations, it is necessary to use a zero-order method.
5. Distributed Algorithms: Many algorithms that we have learned can be made parallel or distributed. One example is Hogwild!, an parallel SGD algorithm that does not use locking.

There are also other related convex problems, where the goal is not simply minimizing an objective. These include saddle point problems and online convex optimization, and have their own algorithms.

Finally, beyond convex optimization, many problems are non-convex, but still important to solve. Sometimes, it is possible to convexify the problem or solve a convex relaxation. However, many important problems such as deep learning, clustering, graphical models, and various combinatorial optimization problems are non-convex optimization problems, for which various non-convex algorithms (and even convex algorithms like Gradient Descent) are used to solve. This creates a number of problems, including escaping from saddle points. There are ways of dealing with each of these problems, but it is an active research area to discover when it is possible to converge to a global optimum solution.