Convex Optimization M2

Lecture 6
Large Scale Optimization
Outline

- First-order methods: introduction
- Exploiting structure
- First order algorithms
  - Subgradient methods
  - Gradient methods
  - Accelerated gradient methods
- Other algorithms
  - Coordinate descent methods
  - Localization methods
  - Franke-Wolfe
  - Dykstra, alternating projection
  - Stochastic optimization
First-order methods: introduction

- Most of these methods are very old (1950-...)
- Very large catalog of algorithms, no unifying theory as in IPM
- Many variations around a few key algorithmic templates
- Better scaling, worst dependence on precision target
- In practice: algorithmic choices are dictated by **problem structure**.

**What subproblem (projection, etc...) can you solve efficiently?**
First Order Algorithms
First-order methods: introduction

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

In theory:

- The theoretical convergence speed of gradient based methods is mostly controlled by the smoothness of the objective.
- Obviously, the geometry of the (convex) feasible set also has an impact.

<table>
<thead>
<tr>
<th>Convex objective ( f(x) )</th>
<th>Iterations. . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nondifferentiable</td>
<td>( O(1/\epsilon^2) )</td>
</tr>
<tr>
<td>Differentiable</td>
<td>( O(1/\epsilon^2) )</td>
</tr>
<tr>
<td>Smooth (Lipschitz gradient)</td>
<td>( O(1/\sqrt{\epsilon}) )</td>
</tr>
<tr>
<td>Strongly convex</td>
<td>( O(\log(1/\epsilon)) )</td>
</tr>
</tbody>
</table>

In practice:

- Compared to IPM, much larger gap between theoretical complexity guarantees and empirical performance.
- Conditioning, well-posedness, etc. also have a very strong impact.
First-order methods: introduction

Solve

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

in \( x \in \mathbb{R}^n \), with \( C \subset \mathbb{R}^n \) convex.

Main assumptions in the subgradient/gradient methods that follow:

- The gradient \( \nabla f(x) \) or a subgradient can be computed efficiently.
- If \( C \) is not \( \mathbb{R}^n \), for any \( y \in \mathbb{R}^n \), the following subproblem can be solved efficiently

\[
\begin{align*}
\text{minimize} & \quad y^T x + d(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

in the variable \( x \in \mathbb{R}^n \), where \( d(x) \) is a strongly convex function.

Typically, \( d(x) = \|x\|_2 \) and this is an Euclidean projection.
Subgradient Method
Subgradient Methods

Subgradient

- Suppose that $f$ is a convex function with $\text{dom} f = \mathbb{R}^n$, and that there is a vector $g \in \mathbb{R}^n$ such that:

  $$f(y) \geq f(x) + g^T(y - x), \quad \text{for all } y \in \mathbb{R}^n$$

- The vector $g$ is called a subgradient of $f$ at $x$, we write $g \in \partial f$.

- Of course, if $f$ is differentiable, the gradient of $f$ at $x$ satisfies this condition.

- The subgradient defines a supporting hyperplane for $f$ at the point $x$. 

Subgradient Methods

Subgradient method:

- Suppose \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is convex
- We update the current point \( x_k \) according to:

\[
x_{k+1} = x_k + \alpha_k g_k
\]

where \( g_k \) is a subgradient of \( f \) at \( x_k \)
- \( \alpha_k \) is the step size sequence
- Similar to gradient descent but, not a descent method . . .
- Instead: use the best point and the minimum function value found so far
Subgradient Methods

Step size strategies:

- Constant step size: $\alpha_k = h$ for all $k \geq 0$
- Constant step length: $\alpha_k / \|g_k\| = h$ for all $k \geq 0$
- Square summable but not summable:
  \[
  \sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty
  \]
- Nonsummable diminishing:
  \[
  \sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \lim_{k \to \infty} \alpha_k = 0
  \]
Subgradient Methods

Convergence:

Assuming $\|g\|_2 \leq G$, for all $g \in \partial f$, we can show

$$f_{\text{best}} - f^* \leq \frac{\text{dist}(x_1, x^*) + G^2 \sum_{i=1}^{k} \alpha_i^2}{2 \sum_{i=1}^{k} \alpha_i}$$

For constant step $\alpha_i = h$, this becomes

$$f_{\text{best}} - f^* \leq \frac{\text{dist}(x_1, x^*)}{2hk} + \frac{G^2 h}{2}$$

to get an $\epsilon$ solution, we set $h = 2\epsilon/G^2$ and

$$\frac{\text{dist}(x_1, x^*)}{2hk} \leq \epsilon$$

hence

$$k \geq \frac{\text{dist}(x_1, x^*) G^2}{4\epsilon^2}.$$
Subgradient Methods

- If the problem has constraints:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in C
\end{align*}
\]

where \( C \subset \mathbb{R}^n \) is a convex set

- Use the Euclidean projection \( p_C(\cdot) \)

\[
x_{k+1} = p_C(x_k + \alpha_k g_k)
\]

- Similar complexity analysis

- Some numerical examples on piecewise linear minimization. . . Problem instance with \( n = 10 \) variables, \( m = 100 \) terms
Constant step length, \( h = 0.05, 0.02, 0.005 \)
Constant step size $h = 0.05, 0.02, 0.005$
Diminishing step rule $\alpha = 0.1/\sqrt{k}$ and square summable step size rule $\alpha = 0.1/k$. 

![Graph showing the comparison between $\alpha = 0.1/\sqrt{k}$ and $\alpha = 0.1/k$]
Constant step length $h = 0.02$, diminishing step size rule $\alpha = 0.1/\sqrt{k}$, and square summable step rule $\alpha = 0.1/k$
Gradient Descent
Gradient descent method

general descent method with $\Delta x = -\nabla f(x)$

given a starting point $x \in \text{dom } f$.
repeat
1. $\Delta x := -\nabla f(x)$.
2. Line search. Choose step size $t$ via exact or backtracking line search.
3. Update. $x := x + t \Delta x$.
until stopping criterion is satisfied.

- stopping criterion usually of the form $\|\nabla f(x)\|_2 \leq \epsilon$
- convergence result: for strongly convex $f$,

$$f(x^{(k)}) - p^? \leq c^k(f(x^{(0)}) - p^?)$$

$c \in (0, 1)$ depends on $m$, $x^{(0)}$, line search type.
- this means $O(\log 1/\epsilon)$ iterations to get $\epsilon$ solution.
- very simple, but often very slow; rarely used in practice
A quadratic problem in $\mathbb{R}^2$

$$f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad (\gamma > 0)$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$:

$$x_1^{(k)} = \gamma \left( \frac{\gamma - 1}{\gamma + 1} \right)^k, \quad x_2^{(k)} = \left( -\frac{\gamma - 1}{\gamma + 1} \right)^k$$

- very slow if $\gamma \gg 1$ or $\gamma \ll 1$
- example for $\gamma = 10$:
Accelerated Gradient Methods
Solve

\[
\begin{align*}
\text{minimize} &\quad f(x) \\
\text{subject to} &\quad x \in C
\end{align*}
\]

in \( x \in \mathbb{R}^n \), with \( C \subset \mathbb{R}^n \) convex.

- **Additional smoothness** assumption: the gradient is Lipschitz continuous

\[
\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \text{for all } x, y \in C
\]

where \( \| \cdot \| \) is the Euclidean norm (to simplify).
Under this new smoothness assumption, we can improve the complexity bound for the most basic gradient method

\[ x_{k+1} = x_k - h \nabla f(x_k) \]

for some \( h > 0 \). We get

\[ f(x_k) - f(x^*) \leq \frac{2L(f(x_0) - f(x^*))\|x_0 - x^*\|^2}{2L\|x_0 - x^*\|^2 + k(f(x_0) - f(x^*))} \]

having set \( h = 1/L \).

Roughly \( O(1/\epsilon) \) iterations to get \( \epsilon \)-solution. This is suboptimal as the lower complexity bound is \( O(1/\sqrt{\epsilon}) \). In what follows, we will see how to reach this optimal complexity.
The fact that the gradient $\nabla f(x)$ is Lipschitz continuous

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \text{for all } x, y \in C$$

has important algorithmic consequences:

- For any $x, y \in \mathbb{R}^n$,

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

  and we get a quadratic upper bound on the function $f(x)$.

- This means in particular that if $y = x - \frac{1}{L}\nabla f(x)$, then

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

  and we get a guaranteed decrease in the function value at each gradient step.
We construct an estimate sequence $\phi_k(x)$ of the function $f(x)$, together with sequences $x_k \in \mathbb{R}^n$ and $\lambda_k \geq 0$, satisfying

$$
\phi_k(x) \leq (1 - \lambda_k)f(x) + \lambda_k \phi_0(x)
$$

and

$$
f(x_k) \leq \phi^*_k \triangleq \min_{x \in \mathbb{R}^n} \phi_k(x).
$$

This means in particular that

$$
f(x_k) - f^* \leq \lambda_k (\phi_0(x^*) - f^*)
$$

(just plug $x^*$ in the inequalities above) so we get convergence if $\lambda_k \to 0$. 
The function $f(x)$ and its estimate functions $\phi_k(x)$:

The functions are $\phi_k(x)$ are increasingly precise approximations of $f(x)$ around the optimum and are easier to minimize.
Intuition behind the method. Use the fact that the gradient is Lipschitz continuous.

- The inequality

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

helps us build the bounds \( \phi_k(x) \).

- In fact, we can pick

\[
\phi_k(x) = \phi^* + \gamma_k \|x - v_k\|^2
\]

for some \( \gamma_k \geq 0 \) and \( v_k \in \mathbb{R}^n \).

- We get the points \( x_{k+1} \) by making a gradient step starting around the minimum of \( \phi_k(x) \) (easy to compute), using the guarantee

\[
f(y) \leq f(x) - \frac{1}{2L} \|\nabla f(x)\|^2
\]
Accelerated Gradient Methods

Also solves minimization problems over simple convex sets $C \subset \mathbb{R}^n$. Define the gradient mapping

$$g_C(y, \gamma) = \gamma(y - x_C(y, \gamma))$$

where

$$x_C(y, \gamma) = \arg\min_{x \in C} \left( f(y) + \nabla f(y)^T (x - y) + \frac{\gamma}{2} \|x - y\|^2 \right)$$

- Here, $g_C(y, \gamma)$ plays the role of the gradient for constrained problems, and satisfies

$$f(x) \geq f(x_C(y, \gamma)) + g_C(y, \gamma)^T (x - y) + \frac{1}{2\gamma} \|g_C(y, \gamma)\|^2 + \frac{\mu}{2} \|x - y\|^2$$

- This means in particular

$$f(x_C(y, \gamma)) \leq f(y) - \frac{1}{2\gamma} \|g_C(y, \gamma)\|^2$$

(just set $y = x$ in the previous inequality).
Accelerated Gradient Methods

Minimize $f(x)$ over $C \subset \mathbb{R}^n$. Assuming $\nabla f(x)$ is Lipschitz continuous with constant $L$ and that $f(x)$ is strongly convex with parameter $\mu \geq 0$.

- Choose $x_0 \in \mathbb{R}^n$ and $\alpha_0 \in (0, 1)$, set $y_0 = x_0$ and $q = \mu / L$.
- For $k = 1, \ldots, k_{\max}$ iterate
  1. Compute $\nabla f(y_k)$ and set
     \[ x_{k+1} = x_C(y_k, \gamma) \]
  2. Compute $\alpha_{k+1} \in (0, 1)$ by solving
     \[ \alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 + q\alpha_{k+1} \]
  3. Update the current point, with
     \[ y_{k+1} = x_{k+1} + \frac{\alpha_k(1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}}(x_{k+1} - x_k) \]
Accelerated Gradient Methods

Suppose we set $\alpha_0 \geq \sqrt{\mu/L}$, we have the following complexity bound

$$f(x_k) - f^* \leq \Delta_0 \min \left\{ \left(1 - \sqrt{\frac{\mu}{L}}\right)^k, \frac{4L}{(2\sqrt{L} + k\sqrt{\gamma_0})^2} \right\}$$

where

$$\Delta_0 = \left(f(x_0) - f^* + \frac{\gamma_0}{2}\|x_0 - x^*\|^2\right) \quad \text{and} \quad \gamma_0 = \frac{\alpha_0(\alpha_0 L - \mu)}{1 - \alpha_0}.$$

When the strong convexity parameter $\mu = 0$, this means roughly $O(1/\sqrt{\epsilon})$ iterations to get an $\epsilon$ solution.

Remarks:

- The iterates $y_k$ are not guaranteed to be feasible (in some case, $f(x)$ is not defined outside of $C$).
- The norm $\| \cdot \|$ is Euclidean. Using other norms is sometimes more efficient.

Both issues can be remedied using an extra minimization subproblem.