Complexity in continuous optimization (5/6)

February 20, 2025

Today: High-order methods

Mathematical Programming (2020) 184:71–120
https://doi.org/10.1007/s10107-019-01406-y

FULL LENGTH PAPER

Series A

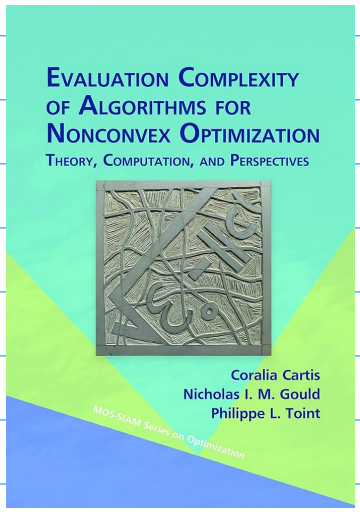
Lower bounds for finding stationary points I

Yair Carmon[1] · John C. Duchi[2] · Oliver Hinder[3] · Aaron Sidford[3]

SIAM J. OPTIM.
Vol. 23, No. 2, pp. 1092–1125

AN ACCELERATED HYBRID PROXIMAL EXTRAGRADIENT METHOD FOR CONVEX OPTIMIZATION AND ITS IMPLICATIONS TO SECOND-ORDER METHODS*

RENATO D. C. MONTEIRO† AND B. F. SVAITER‡



Math. Program., Ser. B (2008) 112:159–181
DOI 10.1007/s10107-006-0089-x

FULL LENGTH PAPER

Accelerating the cubic regularization of Newton's method on convex problems

Yu. Nesterov

HIGH-ORDER OPTIMIZATION

Motivation:

- → we have seen algorithms/complexity for problems with a $C^{1,1}$ objective and $C^{2,2}$ objective
- → We have seen complexity guarantees for
  - first-order optimality ($\|\nabla f(x)\| \leq \varepsilon_1$)
  - second-order optimality ($\|\nabla f(x)\| \leq \varepsilon_1$ and $\lambda_{\min}(\nabla^2 f(x)) \geq -\varepsilon_2$)
  - ⋮

Most of the complexity literature falls into this setting

Q) What can we say when the objective function is $C^{p,p}$ for some $p \in \mathbb{N}^*$?

Yes!

- → Does the use of p^{th}-order derivatives improve complexity with respect to 1st and 2nd order guarantees ? (when $p=2$, we know this is the case for 1st-order guarantees)

Yes in theory but...

- → Can we get stronger optimality guarantees using those derivatives ?

(1) Nonconvex setting

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

$f \in C_L^{p,p}$ for some $L > 0$

$\forall x \in \mathbb{R}^n$, $\nabla^i f(x) \in \mathbb{R}^{\overbrace{n \times n \times \dots \times n}^{i \text{ times}}}$: i^{th} order derivative of f at x

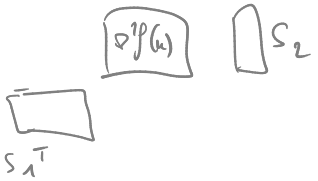
$f \in C_L^{p,p} \Rightarrow \forall (x,s) \in (\mathbb{R}^n)^2,$

$$f(x+s) \le \underbrace{f(x) + \sum_{i=1}^{p} \frac{1}{i!} \nabla^i f(x)[\overbrace{s,\dots,s}^{i \text{ times}}]}_{p\text{th order Taylor expansion of } f \text{ at } x} + \frac{L}{(p+1)!}\|s\|^{p+1}$$

$\nabla^i f(x)[s_1,\dots,s_i]$: $\nabla^i f(x)$ applied to i vectors $s_1,\dots,s_i$

$i=1$ $\quad \nabla^1 f(x)[s_1] \equiv \nabla f(x)^T s_1$

$i=2$ $\quad \nabla^2 f(x)[s_1,s_2] \equiv s_1^T \nabla^2 f(x) s_2 \; (= s_2^T \nabla^2 f(x) s_1)$



$i=3$

$i=1$ $\quad \nabla f(x)[s_1] = \sum_{j=1}^{n} [\nabla f(x)]_j [s_1]_j$

$i=2$ $\quad \nabla^2 f(x)[s_1,s_2] = \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} [\nabla^2 f(x)]_{j_1 j_2} [s_1]_{j_1} [s_2]_{j_2}$

i $\quad \nabla^i f(x)[s_1,\dots,s_i] = \sum_{j_1=1}^{n} \cdots \sum_{j_i=1}^{n} [\nabla^i f(x)]_{j_1 j_2 \cdots j_i} [s_1]_{j_1} \cdots [s_i]_{j_i}$

NB: $n=1$ $\quad f(x+s) \le f(x) + \sum_{i=1}^{p} f^{(i)}(x) s^i + \frac{L}{(p+1)!}|s|^{p+1}$

↳ By analogy with the cases $p=1$ and $p=2$, we can build algorithms based on the p^{th} order Taylor expansion.

Trust-region method

At every iteration k, given (x_k, δ_k), compute

$$f(x_k), \nabla f(x_k), \dots, \nabla^p f(x_k)$$

and s_k as a solution of the subproblem

$$\text{minimize } T_p(x_k, s) \quad \text{subject to } \|s\| \le \delta_k$$

$T_p(x_k, s)$: Model of f around x_k

Trust region (δ_k: radius) → $\|s\| \le \delta_k$

where $T_p(x_k, s) = f(x_k) + \sum_{i=1}^{p} \frac{1}{i!} \nabla^i f(x_k)[s, \dots, s]$

Compute x_{k+1} and δ_{k+1} according to

$$\frac{f(x_k) - f(x_k + s_k)}{T_p(x_k, 0) - T_p(x_k, s_k)}$$

$f(x_k) - f(x_k + s_k)$: change in the true function

$T_p(x_k, 0) - T_p(x_k, s_k)$: change in the model from x_k to $x_k + s_k$

Adaptive regularization (Cubic regularization when $p=2$)

At iteration k, given (x_k, σ_k), compute

$$f(x_k), \dots, \nabla^p f(x_k)$$

and compute s_k as a solution of the subproblem

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \left\{ T_p(x_k, s) + \frac{\sigma_k}{p+1} \|s\|^{p+1} \right\}$$

→ unconstrained problem
→ Regularization $\|s\|^{p+1}$ guarantees that there exist a solution

Compute x_{k+1} and σ_{k+1} according to

$$\frac{f(x_k) - f(x_k + s_k)}{T_p(x_k, 0) - T_p(x_k, s_k)}$$

→ Both TR (Trust Region) and AR (Adaptive Regularization) only differ from the case $p=2$ in the definition of the subproblems (and the number of derivatives they use)

Complexity guarantees

(Basic) TR: Reaches x such that $\|\nabla f(x)\| \leq \underbrace{\varepsilon_1}_{\in (0,1)}$ in at most $O(\varepsilon_1^{-(p+1)})$ iterations

Before: $p=2$, $O(\varepsilon_1^{-2})$

Worse guarantee than a second-order method for $p>2$

Related to the decrease guarantee in TR methods:

$p=1$ If $\|\nabla f(x_k)\| \geq \varepsilon_1$, then

$$T_2(x_k, 0) - T_2(x_k, s_k) \geq O(\|\nabla f(x_k)\|^2) \geq O(\varepsilon_1^2)$$

decrease for a step $s = -\nabla f(x_k)$

$p>2$ If $\|\nabla f(x_k)\| \geq \varepsilon_1$

$$T_p(x_k,0) - T_2(x_k,s_k) \geq \min_{0\leq t\leq \frac{\delta_k}{\|\nabla f(x_k)\|}} \left(T_p(x_k,0) - T_p(x_k, -t\nabla f(x_k))\right)$$

$$\geq O(\|\nabla f(x_k)\|^{p+1})$$

$$\geq O(\varepsilon_1^{(p+1)})$$

($p=2$, If $\|\nabla f(x_k)\| \geq \varepsilon_1$

$$T_p(x_k,0) - T_2(x_k,s_k) \geq O(\|\nabla f(x_k)\|^2 \varepsilon_1, \varepsilon_1^3)$$

$$\geq O(\varepsilon_1^3)$$

AR:

Reaches x_k such that $\|\nabla f(x_k)\| \leq \varepsilon_1$ in at most

$$O(\varepsilon_1^{-\frac{(p+1)}{p}}) \text{ iterations}$$

.

$p=1$ ε_1^{-2}

$p=2$ $\varepsilon_1^{-3/2}$

$p=3$ $\varepsilon_1^{-4/3}$

$$\varepsilon^{-\frac{p+1}{p}} \xrightarrow[p\to\infty]{} \varepsilon^{-1}$$

Analogous: At every iteration,

$$\to T_p(x_k,0) - T_p(x_k,s_k) \geq O(\|s_k\|^{p+1})$$

$$\to \|s_k\| \geq O(\|\nabla f(x_{k+1})\|^{1/p})$$

Lower bounds (Carmon et al 2020)

Generic analysis

- Problem class $\mathcal{F} = \{ f : \mathbb{R}^d \to \mathbb{R}$ for some $d \geq 1, f \in C_L^{p,p}, f(x_0) - \inf_x f(x) \leq \Delta \}$

  f orthogonally invariant

- Algorithmic class $\mathcal{A}$

  $x_0 = 0_{\mathbb{R}^n}$ (without loss of generality)

  $\forall k \geq 1 \quad x_k = A_k\big(f(x_0), \nabla f(x_0), \dots, \nabla^p f(x_0), f(x_{k-1}), \nabla f(x_{k-1}), \dots, \nabla^p f(x_{k-1})\big)$

  A_k: mapping to $\mathbb{R}^n$

$\forall a \in \mathcal{A}, \forall f \in \mathcal{F}, \quad \forall \varepsilon_1 > 0$

$$T_{\varepsilon_1}(a, f) = \inf \{ k \mid \|\nabla f(x_k)\| \leq \varepsilon_1 \}$$

Th 1 $\forall \varepsilon_1 > 0$

$$\inf_{a \in \mathcal{A}} \sup_{f \in \mathcal{F}} T_\varepsilon(a, f) \leq C \Delta L^{1/p} \varepsilon^{-\frac{p+1}{p}}$$

$C > 0$

and

$$\inf_{a \in \mathcal{A}} \sup_{f \in \mathcal{F}} T_\varepsilon(a, f) \geq \hat{C} \Delta (L/\varepsilon)^{1/p} \varepsilon^{-\frac{p+1}{p}}$$

$n = O(p \log p)$

Orthogonal invariance of $\mathcal{F}$

$\forall f \in \mathcal{F}$, $f : \mathbb{R}^d \to \mathbb{R}$ and $\forall U \in \mathbb{R}^{d' \times d}$, $U^T U = I_d$,

$f' : \mathbb{R}^{d'} \to \mathbb{R}$, $x \mapsto f(U^T x) \in \mathcal{F}$

Because of orthogonal invariance,

$$\inf_{a \in A} \sup_{f \in \mathcal{F}} T_\varepsilon(a, f) \geq \inf_{a \in A^{zr}} \sup_{f \in \mathcal{F}} T_\varepsilon(a, f)$$

$A^{zr} \subseteq A$: "zero-respecting algorithms"

Inspiration : Nesterov's example for accelerated gradient

$\Rightarrow$ Any algorithm would only update 1 coordinate of the iterate per iteration

$\Rightarrow$ By design :

$\rightarrow$ Start with $x_0 = \begin{bmatrix} 0 \\ ! \\ 0 \end{bmatrix}$

$x^* = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $\rightarrow$ It 1 : $x_1 = \begin{bmatrix} [x^*]_1 \\ 0 \\ ! \\ 0 \end{bmatrix}$

$\rightarrow$ It k : $x_k = \begin{bmatrix} [x^*]_1 \\ \vdots \\ [x^*]_k \\ 0 \\ ! \\ 0 \end{bmatrix}$

A zero-respecting algorithm satisfies ($x_0 = \begin{bmatrix} 0 \\ ! \\ 0 \end{bmatrix}$)

$$\forall k \geq 1, \quad \operatorname{supp}(x_k) \subseteq \bigcup_{1 \leq q \leq p} \bigcup_{j < k} \operatorname{supp}\left(\nabla_q f(x_j)\right)$$

where $supp(\cdot)$ denotes the support

$$\forall x \in \mathbb{R}^n, \quad supp(x) = \{ i \mid 1 \le i \le n, \ x_i \neq 0 \}$$

Recursive definition of the support:

$$\forall T \in \mathbb{R}^{\overbrace{n \times \dots \times n}^{j \text{ times}}}, \quad \forall i = 1..n, \text{ define } T^{(i)} \in \mathbb{R}^{\overbrace{n \times \dots \times n}^{j-1}}$$

$$\left[T^{(i)}\right]_{m_1, \dots, m_{j-1}} = T_{i, m_1, \dots, m_{j-1}}$$

$$supp(T) = \max_{1 \le i \le n} supp(T_i)$$

→ A zero-respecting algorithm does not update coordinates of the iterate that correspond to zero coordinates

Zero-respecting function

$f \in \mathcal{F}$ is zero-respecting if

$\forall x$ such that $supp(x) \subseteq \{1, \dots, i-1\}$, $\left(x = \begin{bmatrix} x_1 \\ \vdots \\ x_{i-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right)$

$$\bigcup_{0 \le q \le p} supp\left(\nabla^q f(x) \right) \subseteq \{1, \dots, i\}$$

Proof technique: Build a zero-respecting function such that any zero-respecting algorithm will take at least $\Omega\left(\varepsilon_1^{-\frac{p+1}{p}}\right)$ iterations to satisfy $\|\nabla f(x)\| \le \varepsilon_1$

Function construction

$\forall T \geq 1$, $\bar{f}_T : \mathbb{R}^T \longrightarrow \mathbb{R}$

$$x \longmapsto -\psi(1)\phi(x_1) + \sum_{i=2}^{T} \left[\psi(-x_{i-1})\,\Phi(-x_i) - \psi(x_{i-1})\,\Phi(x_i) \right]$$

$$\psi(x) = \begin{cases} \exp\left(1 - \frac{1}{(2x-1)^2}\right) & \text{if } x > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$\Phi(x) = \sqrt{e} \int_{-\infty}^{x} e^{-t^2/2}\, dt$$

Can show that $\bar{f}_T \in C_l^{p,p}$ with $l \leq \exp\left(\frac{T}{2}\log p + p\right) = O(p^p \exp p)$

Then, define $f_{\varepsilon_1} : \mathbb{R}^{T_{\varepsilon_1}} \longrightarrow \mathbb{R}$

$$x \longmapsto \frac{L}{l}\left(\frac{l\,\varepsilon_1}{L}\right)^{\frac{p+1}{p}} \bar{f}_{T_{\varepsilon_1}}\left(\left(\frac{L}{l\varepsilon_1}\right)^{\frac{1}{p}} x\right)$$

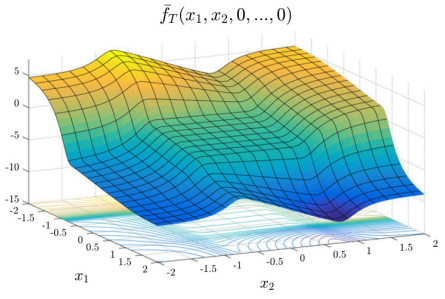
$$T_{\varepsilon_1} = \left\lfloor \frac{\Delta\, L^{1/p}}{12\, l^{1/p}}\, \varepsilon^{-\frac{(p+1)}{p}} \right\rfloor$$

f_{ε_1} is $C_L^{p,p}$, $f \in F$

and

$\forall x \in \mathbb{R}^{T_{\varepsilon_1}}$ such that

$x_{T_{\varepsilon_1}} = 0$, $\|\nabla f_{\varepsilon_1}(x)\| > \varepsilon_1$



→ If the algorithm only updates coordinate i at iteration i, it takes at least T_{ε_1} iterations to satisfy $\|\nabla f_{\varepsilon_1}(x)\| \leq \varepsilon_1$

Current challenge: Implement the AR method for $p \geq 3$

↳ Lack of good method for solving the subproblem $\underset{s \in \mathbb{R}^n}{\text{minimize}}\ T_p(x,s) + \frac{\sigma}{p+1}\|s\|^{p+1}$

↳ Believed to be impractical for $p \geq 4$, the case $p=3$ is subject of current research

(2) the convex setting

Starting point: $\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x)$, $f \in C^{2,2}$

Adaptive cubic regularization ($p=2$): $O(\varepsilon^{-3/2})$ complexity

If f is convex, for the same algorithm: $O(\varepsilon^{-1/2})$ complexity

same complexity than accelerated gradient, but only known to be optimal for $C^{1,1}$ convex functions

Nesterov (2008):
For $f \in C^{2,2}$ convex, an accelerated variant of cubic regularization has complexity $O(\varepsilon^{-1/3})$

Monteiro and Svaiter (2013)

Their algorithm uses regularized Newton steps and for $C^{2,2}$ convex functions, has complexity $O(\varepsilon^{-2/7})$

$\Rightarrow$ this is the optimal complexity for $C^{2,2}$ convex functions

$\frac{2}{7} < \frac{2}{6} = \frac{1}{3}$

$\Rightarrow$ Two families of algorithms

- Optimal methods $O(\varepsilon^{-2/7})$
- Near-optimal methods $O(\varepsilon^{-1/3})$

$\Rightarrow$ Around 2018, the results were extended to high-order for $C^{p,p}$ convex

Near-optimal : $O\left(\varepsilon^{-\frac{1}{p+1}}\right)$

Optimal : $O\left(\varepsilon^{-\frac{2}{3p+1}}\right)$

$p=1$ $\quad \varepsilon^{-\frac{1}{2}} = \varepsilon^{-\frac{2}{4}}$

$p=2$ $\quad \varepsilon^{-1/3}$ vs $\varepsilon^{-2/7}$

Lower bound : $O\left(\varepsilon^{-\frac{2}{3p+1}}\right)$ (2017)

The main results in that area appeared in COLT (Conference on Learning Theory)

(4) High-order conditions

1st order condition: If x minimum of f (C^1), then

$$\|\nabla f(x)\| = 0$$

2nd order — : ——— C^2 then

$$\|\nabla f(x)\| = 0$$
$$\nabla^2 f(x) \geq 0$$

3rd order — : ——— C^3 then

$$\|\nabla f(x)\| = 0$$
$$\nabla^1 f(x) \geq 0$$
$$\nabla^3 f(x)[s,s,s] = 0$$
$$\forall s,\ \nabla^2 f(x)s = 0$$

⋮

p^{th} order

4^{th} order condition $\equiv$ Checking copositivity (Hard)