A VU-Point of View of Nonsmooth Optimization

Claudia Sagastizábal
IMECC-UNICAMP BRAZIL, adjunct researcher
sagastiz@unicamp.br

ENSTA, January 14th and 15th, 2019

(supported by ENSTA, PGMO, and Fondation X)

Program

1. Yesterday morning:
   Introduction to nonsmooth convex optimization
2. Yesterday afternoon:
   Models and the proximal point algorithm
3. Today morning:
   Bundle methods and the Moreau-Yosida regularization
4. Today afternoon:
   Beyond first order: VU-decomposition methods

Introduction to nonsmooth convex optimization

Let's start with a question

What is the

fast*est* method

you know

to solve $F(x) = 0$,

a nonlinear system of equations?

Answer provided by Isaac Newton

source: Alan Rust

Answer: Newton's method

from J-Ch. Gilbert's classes

$$\begin{aligned} 0 &= F(x^*) \\ &\approx F(x^k) + F'(x^k)d \end{aligned}$$

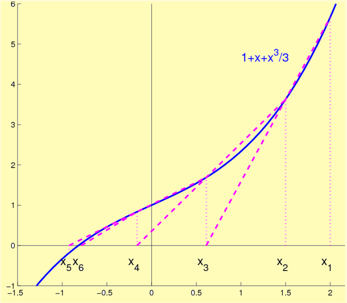
fast **convergence** $x^{k+1} = x^k + d^*$

Answer: Newton's method

from J-Ch. Gilbert's classes

$$\begin{aligned} 0 &= F(x^*) \\ &\approx F(x^k) + F'(x^k)d \end{aligned}$$

fast convergence $x^{k+1} = x^k + d^*$



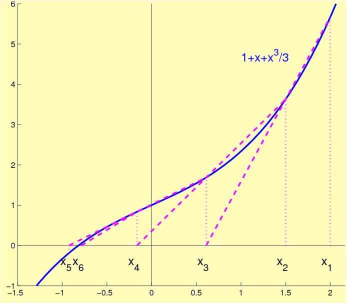
Answer: Newton's method

from J-Ch. Gilbert's classes

$$
\begin{aligned}
0 &= F(x^*) \\
&\approx F(x^k) + F'(x^k)d
\end{aligned}
$$

fast **convergence** $x^{k+1} = x^k + d^*$

one linear system per iteration



Newton method is accurate

$$F(x) = 1 + x + x^3/3$$

1	2.00000000000000	0
2	0.86666666666667	1
3	-0.32323745064862	1
4	-0.92578663808031	1
5	-0.82332584261905	2
6	-0.81774699537697	5
7	-0.81773167400186	9
8	-0.81773167388682	15

Newton

From J-Ch. Gilbert's classes

Le résultat de base :

Si · $F(x_*) = 0$,

· F est $C^{1,1}$ dans un voisinage de x_*,

· $F'(x_*)$ est inversible,

alors il existe un voisinage V de x_* tel que si $x_1 \in V$, l'algorithme de Newton est bien défini et génère une suite $\{x_k\} \subset V$ qui converge *quadratiquement* vers x_* : il existe une constante C telle que

$$\|x_{k+1} - x_*\| \leq C\|x_k - x_*\|^2, \quad \forall k \geq 1.$$

- Si F est seulement C^1, la convergence n'est que superlinéaire :

$$\frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|} \to 0, \quad \text{pour } k \to \infty.$$

And Newton's method for optimization?

$$\begin{aligned} 0 &= F(x^*) \\ &\approx F(x^k) + F'(x^k)d \end{aligned}$$

fast **convergence**

And Newton's method for optimization?

$$\begin{aligned} 0 &= F(x^*) \\ &\approx F(x^k) + F'(x^k)d \end{aligned}$$

fast **convergence**

In optimization

$$F(x) = \nabla f(x)$$

for an objective f

And Newton's method for optimization?

$$
\begin{aligned}
0 &= \nabla f(x^*) \\
&\approx \nabla f(x^k) + \nabla^2 f(x^k) d
\end{aligned}
$$

fast **convergence**

In optimization

$$F(x) = \nabla f(x)$$

$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1} \nabla f(x^k)$$

for an objective f

And Newton's method for optimization?

$$\begin{aligned} 0 &= \nabla f(x^*) \\ &\approx \nabla f(x^k) + \nabla^2 f(x^k) d \end{aligned}$$

fast **convergence**

$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1} \nabla f(x^k)$$

In optimization

$$F(x) = \nabla f(x)$$

for an objective f

$$\min f \approx \min f\textbf{-model}$$

$$\min_d f(x^k) + \langle \nabla f(x^k), d \rangle + \frac{1}{2} \langle \nabla^2 f(x^k) d, d \rangle$$

From J-Ch. Gilbert's classes

- Convergence :

Si $\cdot$ $\nabla f(x_*) = 0$,

$\cdot$ f est $C^{2,1}$ dans un voisinage de x_*,

$\cdot$ $\nabla^2 f(x_*)$ est inversible,

alors il existe un voisinage V de x_* tel que si $x_1 \in V$, l'algorithme de Newton est bien défini et génère une suite $\{x_k\} \subset V$ qui converge *quadratiquement* vers x_*.

From J-Ch. Gilbert's classes

- Convergence :

  Si · $\nabla f(x_*) = 0$,

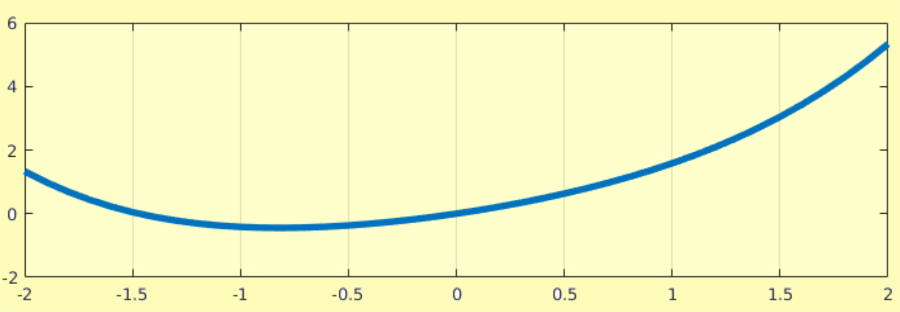
  · f est $C^{2,1}$ dans un voisinage de x_*,

  · $\nabla^2 f(x_*)$ est inversible,

  alors il existe un voisinage V de x_* tel que si $x_1 \in V$, l'algorithme de Newton est bien défini et génère une suite $\{x_k\} \subset V$ qui converge *quadratiquement* vers x_*.

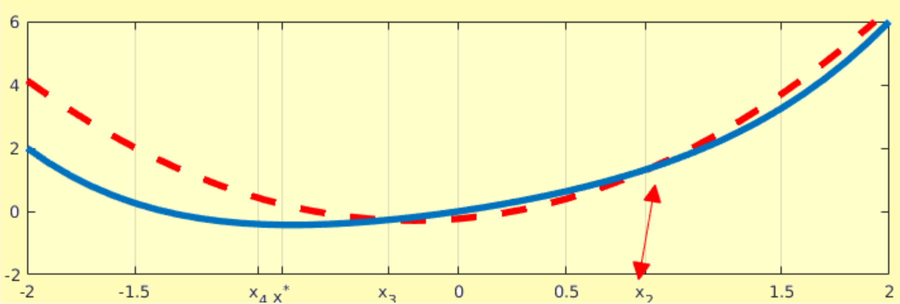
- $f \in C^2 \Longrightarrow$ superlinear convergence

Newton iterates for optimization

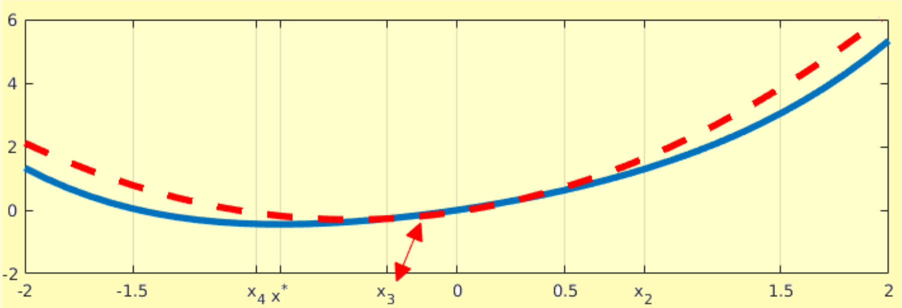


$$F(x) = 1 + x + x^3/3 \quad \Longrightarrow f(x) = x + x^2/2 + x^4/12$$

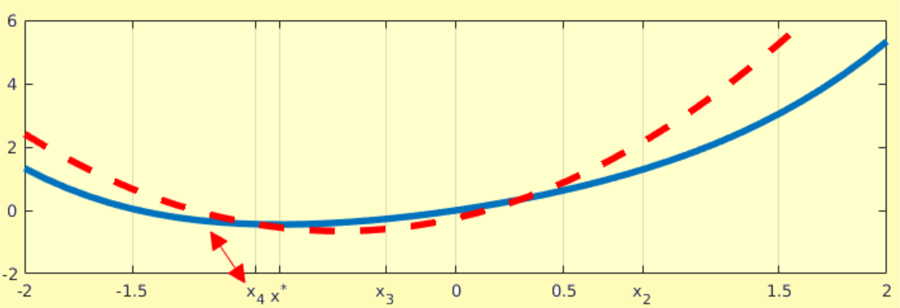
Newton iterates for optimization



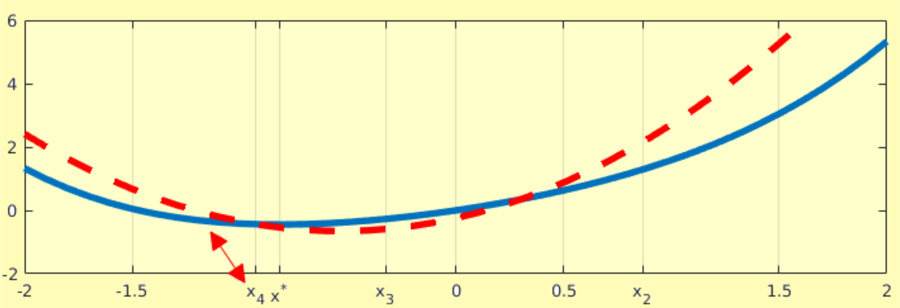
Newton iterates for optimization



Newton iterates for optimization

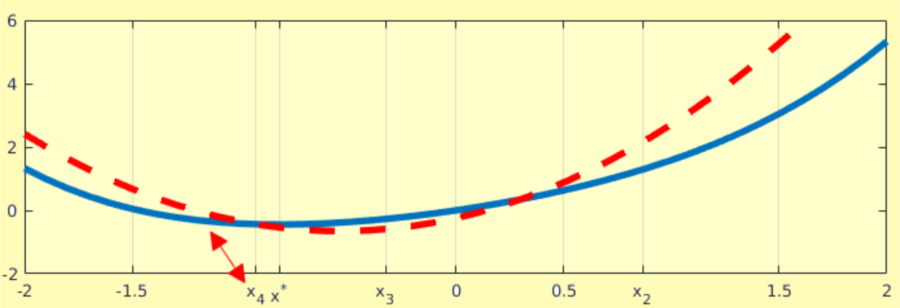


Newton iterates for optimization



Can we avoid computing the Hessian matrix?

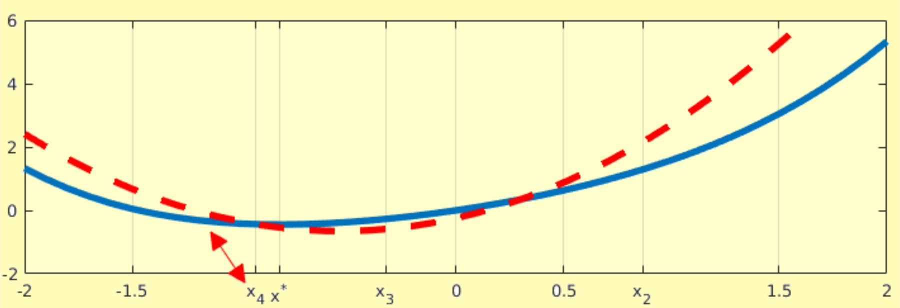
Newton iterates for optimization



Can we avoid computing the Hessian matrix? **YES!**

$$\min_{d} f(x^k) + \left\langle \nabla f(x^k), d \right\rangle + \frac{1}{2} \left\langle \boxed{M^k} d, d \right\rangle$$

Newton iterates for optimization

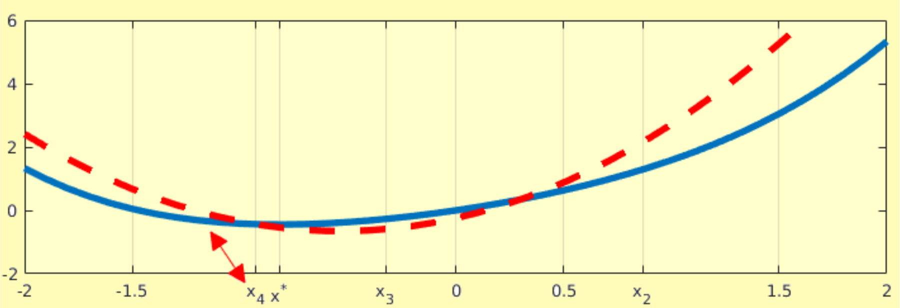


Can we avoid computing the Hessian matrix? **YES!**

$$\min_d f(x^k) + \left\langle \nabla f(x^k), d \right\rangle + \frac{1}{2} \left\langle \boxed{M^k} d, d \right\rangle$$

quasi-Newton matrix

Newton iterates for optimization



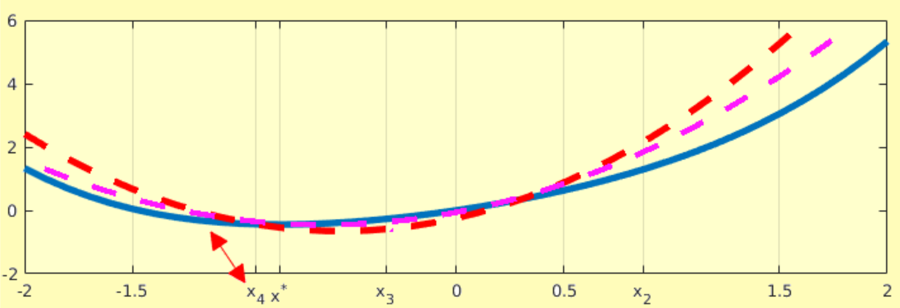
Can we avoid computing the Hessian matrix? **YES!**

$$\min_d f(x^k) + \left\langle \nabla f(x^k), d \right\rangle + \frac{1}{2} \left\langle M^k d, d \right\rangle$$

quasi-Newton matrix

$$0 = \nabla f(x^k) + M^k d$$

quasi-Newton iterates for optimization



Eventually, the true Hessian curvature is estimated only along the generated directions (secant equation)

quasi-Newton methods are **accurate** too!

1	2.00000000000000	0
2	1.50000000000000	0
3	0.61224489795918	1
4	-0.16202797536640	1
5	-0.92209500449059	1
6	-0.78540447895661	1
7	-0.81609056319699	3
8	-0.81775774021392	5
9	-0.81773165292101	8
10	-0.81773167388656	13
11	-0.81773167388682	15

quasi-Newton

1	2.00000000000000	0
2	0.86666666666667	1
3	-0.32323745064862	1
4	-0.92578663808031	1
5	-0.82332584261905	2
6	-0.81774699537697	5
7	-0.81773167400186	9
8	-0.81773167388682	15

Newton

Dennis & Moré Criterion

from J-Ch. Gilbert's classes

IF

- $\{x^{k+1} = x^k + M^k d^k\}$ **converges to** x^*
- $\nabla^2 f(x^*)$ **is non singular**
- $\nabla f(x^*) = 0$

Dennis & Moré Criterion

from J-Ch. Gilbert's classes

IF

- $\{x^{k+1} = x^k + M^k d^k\}$ **converges to** x^*
- $\nabla^2 f(x^*)$ **is non singular**
- $\nabla f(x^*) = 0$

THEN

the convergence is superlinear if and only if

$$\Big(M^k - \nabla^2 f(x^*)\Big)(x^{k+1} - x^k) = o(\|x^{k+1} - x^k\|)$$

Dennis & Moré Criterion

from J-Ch. Gilbert's classes

IF

- $\{x^{k+1} = x^k + M^k d^k\}$ **converges to** x^*
- $\nabla^2 f(x^*)$ **is non singular**
- $\nabla f(x^*) = 0$

THEN

the convergence is superlinear if and only if

$$\left(M^k - \nabla^2 f(x^*)\right)(x^{k+1} - x^k) = o(\|x^{k+1} - x^k\|)$$

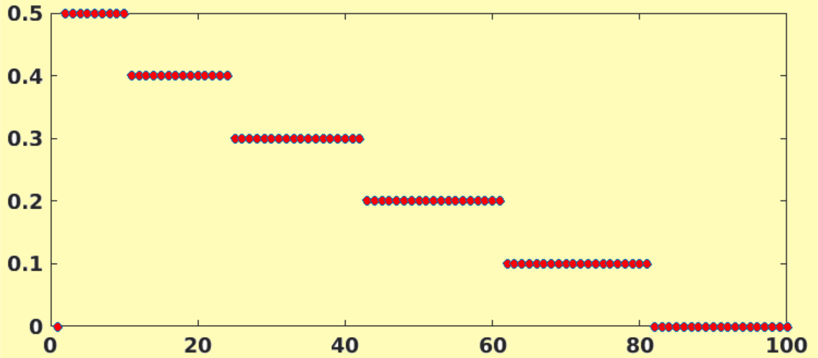
Eventually, the true Hessian curvature is estimated only along the generated directions (secant equation)

What about 1st-order methods?

The same game, this time using a **gradient** method $x^{k+1} = x^k - t_k \nabla f(x^k) = x^k - t_k F(x^k)$ for $t_k > 0$ a sufficiently small stepsize (note easy calculation)

What about 1st-order methods?

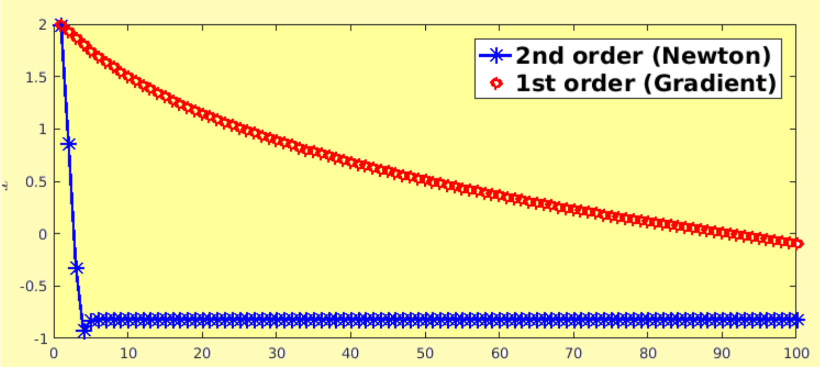
The same game, this time using a **gradient** method $x^{k+1} = x^k - t_k \nabla f(x^k) = x^k - t_k F(x^k)$ for $t_k > 0$ a sufficiently small stepsize (note easy calculation)



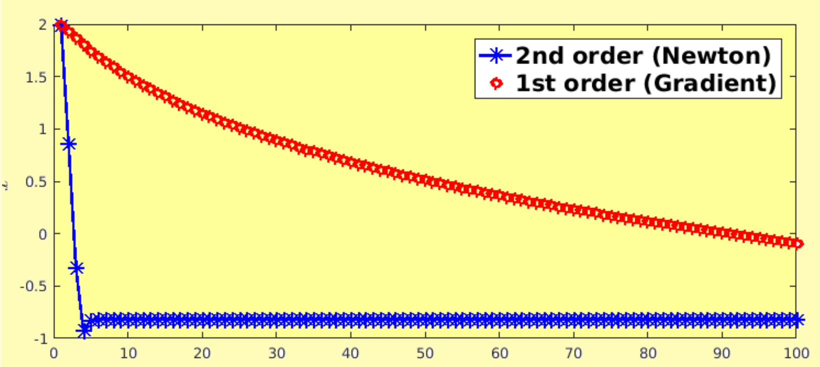
$x^{100} = -0.088237488857720$

($x^* = -0.81773167386824$)

Comparison of x-values



Comparison of x-values



Regarding F-values,

- **with gradient method, after 100 iterations,** $F(x^100) = 0.9115$
- **with Newton's method, after 9 iterations,** $F(x^9) = -8.0491 \times 10^{-16}$

Take away message from the smooth world

- Newton-like methods are *fast*

Take away message from the smooth world

- Newton-like methods are *fast*
- *fast* means accurate (# of digits)

Take away message from the smooth world

- Newton-like methods are fast
- fast means accurate (# of digits)
- Accuracy reached by using a “more than 1st-order” **model** for f

Take away message from the smooth world

- Newton-like methods are *fast*
- *fast* means accurate (# of digits)
- Accuracy reached by using a "more than 1st-order" **model** for f
- No need to approximate the Hessian everywhere

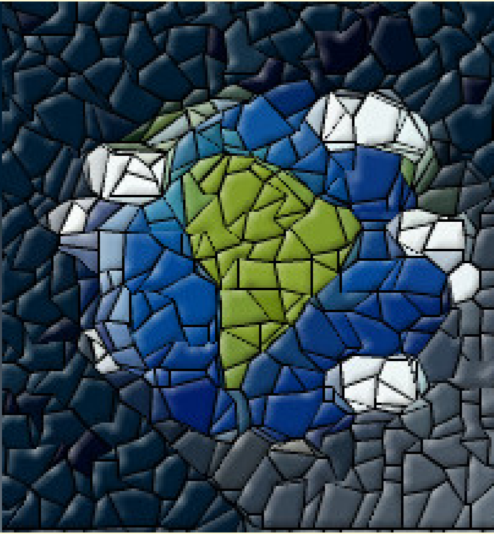
Take away message from the smooth world

- Newton-like methods are *fast*
- *fast* means accurate (# of digits)
- Accuracy reached by using a "more than 1st-order" **model** for f
- No need to approximate the Hessian everywhere

for functions that are C^2

and have an invertible Hessian at x^*

Moving to the nonsmooth world



Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points Algorithms defined according on how much information is provided by certain *oracle*

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle*

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* an **informative oracle**

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle*

an **informative oracle**

x

$f(x)$

the full $\partial f(x)$

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **“black-box”**

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **“black-box”**

x

$f(x)$

ONE $g(x) \in \partial f(x)$

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **“black-box”**

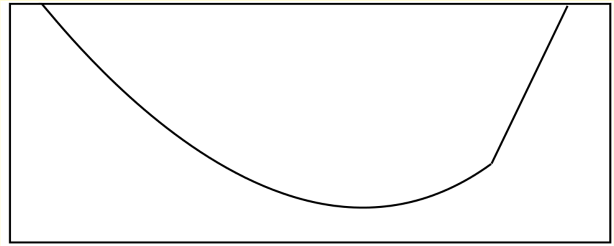
x

$f(x)$

ONE $g(x) \in \partial f(x)$

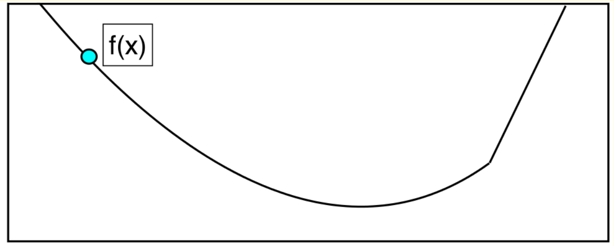
A quick overview of Convex Analysis

An example of a convex nonsmooth function



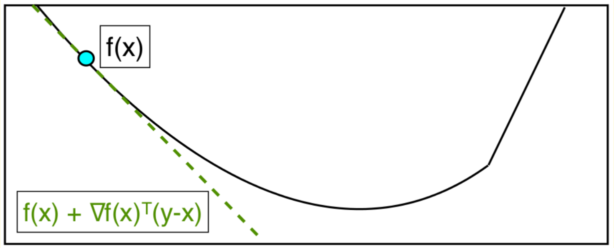
A quick overview of Convex Analysis

An example of a convex nonsmooth function



A quick overview of Convex Analysis

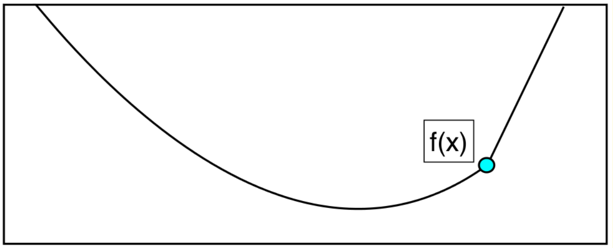
An example of a convex nonsmooth function



$$\begin{aligned}\partial f(x) &= \{\nabla f(x)\} \\ &= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}\end{aligned}$$

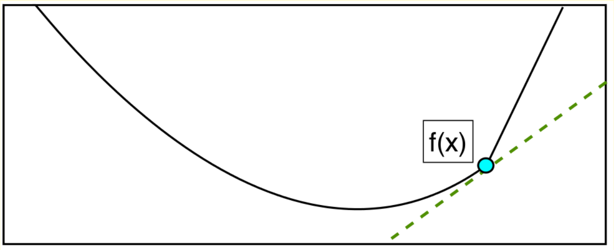
A quick overview of Convex Analysis

An example of a convex nonsmooth function



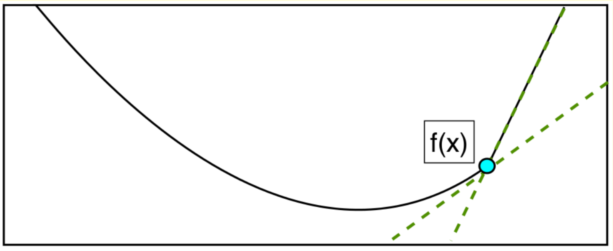
A quick overview of Convex Analysis

An example of a convex nonsmooth function



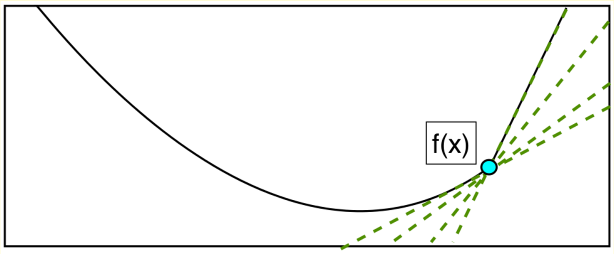
A quick overview of Convex Analysis

An example of a convex nonsmooth function



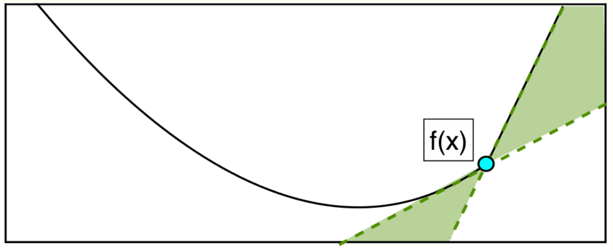
A quick overview of Convex Analysis

An example of a convex nonsmooth function



A quick overview of Convex Analysis

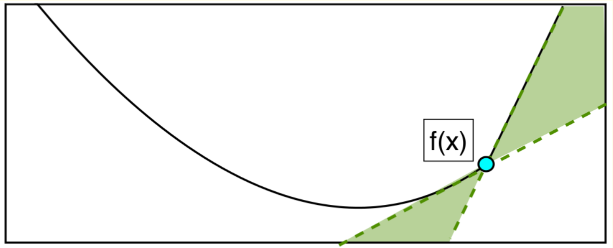
An example of a convex nonsmooth function



$$\partial f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\}$$

A quick overview of Convex Analysis

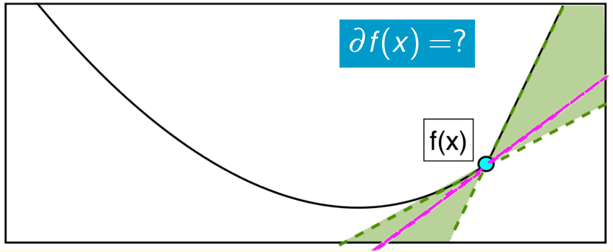
An example of a convex nonsmooth function



$$
\begin{aligned}
\partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\} \\
&= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}
\end{aligned}
$$

A very useful calculus rule for subdifferentials

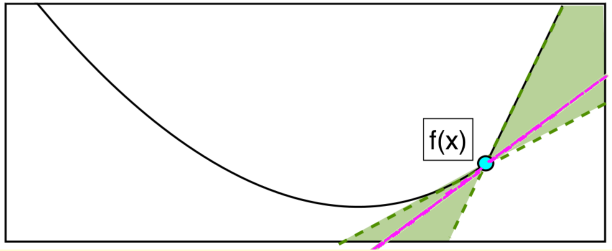
A finite max-function $f(x) := \max_{j \in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex and differentiable



$$\begin{aligned}\partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\} \\ &= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}\end{aligned}$$

A very useful calculus rule for subdifferentials

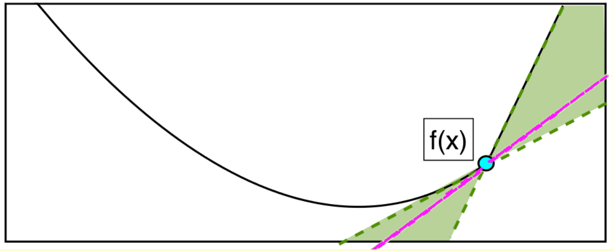
A finite max-function $f(x) := \max_{j \in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex and differentiable



$$\begin{aligned} \partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\} \\ &= \boxed{conv\{\nabla f^j(x) : j \in I(x)\}} \end{aligned}$$

A very useful calculus rule for subdifferentials

A finite max-function $f(x) := \max_{j\in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex and differentiable



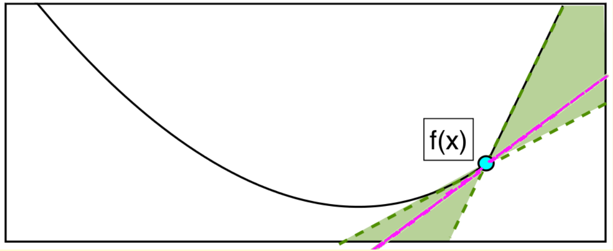
only active indices!

$$\begin{aligned} \partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y-x) \text{ for all y}\} \\ &= conv\{\nabla f^j(x) : j \in I(x)\} \end{aligned}$$

$$I(x) := \left\{ j \in I : f(x) = f^j(x) \right\}$$

A very useful calculus rule for subdifferentials

A finite max-function $f(x) := \max_{j \in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex and differentiable



only active indices!

$$\partial f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\}$$

$$= conv\{\nabla f^j(x) : j \in I(x)\}$$

$$I(x) := \left\{j \in I : f(x) = f^j(x)\right\}$$

What is the subdifferential of $f(x) = |x|$?

Extension to compact set I and nonsmooth f^j

A sup-function $f(x) := \sup_{j\in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex,

- I compact
- $j \mapsto f^j(x)$ is upper-semicontinuous

$$\partial f(x) = cl\Big(conv \cup_{j\in I(x)} \partial f^j(x)\Big)$$

where the active set is defined as before:

$$I(x) := \left\{ j \in I : f(x) = f^j(x) \right\}$$

Extension to compact set I and nonsmooth f^j

A sup-function $f(x) := \sup_{j \in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex,

- I compact
- $j \mapsto f^j(x)$ is upper-semicontinuous

$$\partial f(x) = cl\Big(conv \cup_{j \in I(x)} \partial f^j(x) \Big)$$

where the active set is defined as before:

$$I(x) := \left\{ j \in I : f(x) = f^j(x) \right\}$$

Given n symmetric matrices of order m, consider

$$A(x) := x_1 A_1 + x_2 A_2 + \ldots x_n A_n$$

Extension to compact set I and nonsmooth f^j

A sup-function $f(x) := \sup_{j \in I} f^j(x)$, with $f^j : \mathbb{R}^n \to \mathbb{R}^n$ convex,

- I compact
- $j \mapsto f^j(x)$ is upper-semicontinuous

$$\partial f(x) = cl\Big(conv \cup_{j \in I(x)} \partial f^j(x)\Big)$$

where the active set is defined as before:

$$I(x) := \left\{ j \in I : f(x) = f^j(x) \right\}$$

Given n symmetric matrices of order m, consider

$A(x) := x_1 A_1 + x_2 A_2 + \ldots x_n A_n$

What is the subdifferential of $f(x) = \lambda_{\max}(A(x))$?

(the maximum-eigenvalue)

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **“black-box”**

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **“black-box”**

x

$f(x)$

ONE $g(x) \in \partial f(x)$

Computational NSO: what do we mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points. Algorithms defined according on **how much** information is provided by certain *oracle* a **"black-box"**

x

$f(x)$

ONE $g(x) \in \partial f(x)$

How common are nonsmooth objective functions?

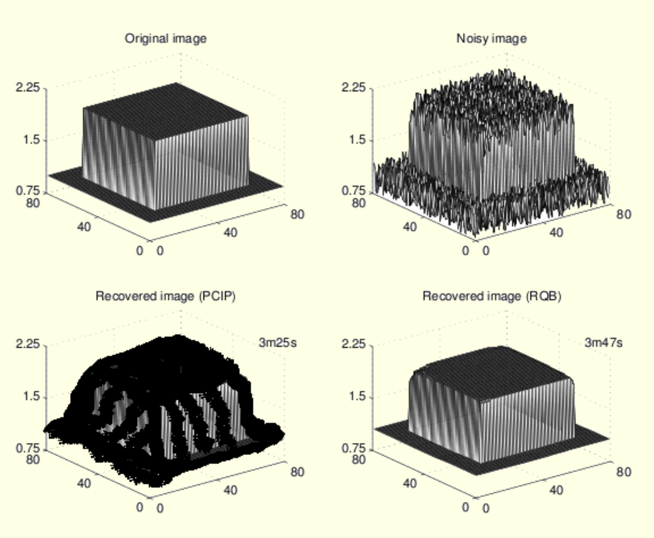
When does nonsmoothness appear?

- * if the **nature** of the problem imposes a nonsmooth model; or
- * if **sparsity** of the solution is a concern; or
- * in problems difficult to solve,
  - because they are large scale
  - because they are heterogeneous

sometimes the **solution method** induces nonsmoothness

Example of NS model

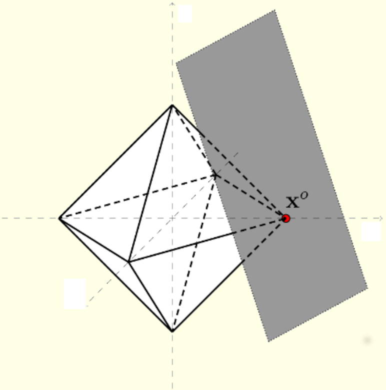
Recovery of **blocky** images (ℓ_1-regularization of TV)



Example of sparse optimization

$$\min\{\|x\|_1 : Ax = b\}$$

Basis pursuit: find least 1-norm point on the affine plane
Tends to return a sparse point (sometimes, the sparsest)



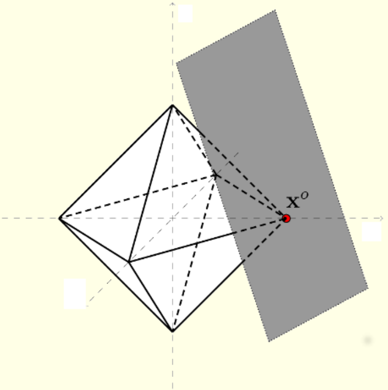
ℓ_1 ball touches the affine plane

Example of sparse optimization

$$\min\{\|x\|_1 : Ax = b\}$$

Basis pursuit: find least 1-norm point on the affine plane

Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the affine plane

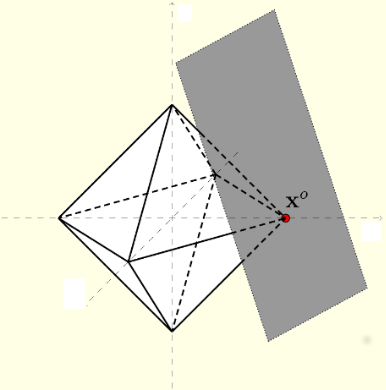
LASSO denoises basis pursuit

$$\min\left\{\|Ax - b\|_2^2 : \|x\|_1 \le \tau\right\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : Ax = b\}$$

Basis pursuit: find least 1-norm point on the affine plane
Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the affine plane

LASSO denoises basis pursuit

$$\min\left\{\|Ax-b\|_2^2 : \|x\|_1 \leq \tau\right\}$$

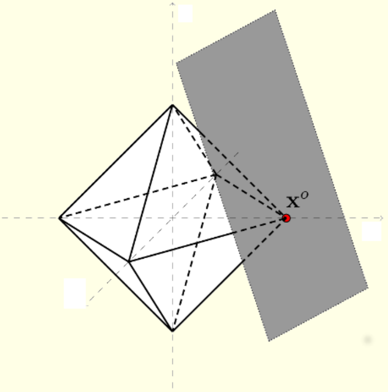
or

$$\min\left\{\|x\|_1 + \frac{\mu}{2}\|Ax-b\|_2^2\right\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : Ax = b\}$$

Basis pursuit: find least 1-norm point on the affine plane
Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the affine plane

LASSO denoises basis pursuit

$$\min\left\{\|Ax - b\|_2^2 : \|x\|_1 \le \tau\right\}$$

or

$$\min\left\{\|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2\right\}$$

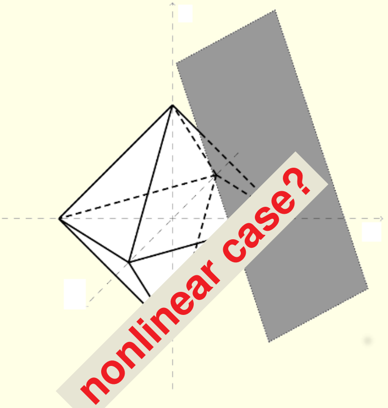
or

$$\min\left\{\|x\|_1 : \|Ax - b\|_2^2 \le \sigma\right\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : Ax = b\}$$

Basis pursuit: find least 1-norm point on the affine plane
Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the affine plane

LASSO denoises basis pursuit

$$\min\left\{\|Ax-b\|_2^2 : \|x\|_1 \le \tau\right\}$$

or

$$\min\left\{\|x\|_1 + \frac{\mu}{2}\|Ax-b\|_2^2\right\}$$

or

$$\min\left\{\|x\|_1 : \|Ax-b\|_2^2 \le \sigma\right\}$$

Example of sparse optimization

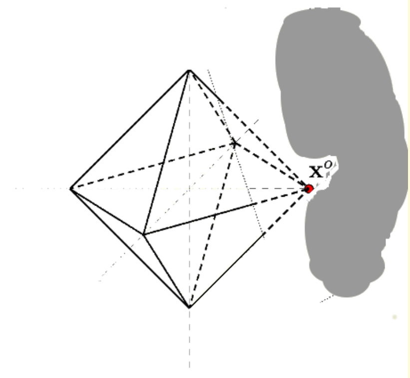
$$\min\{\|x\|_1 : \boxed{h(x) \leq b}\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : \boxed{h(x) \le b}\}$$

Basis pursuit: find least 1-norm point on a **nonlinear set**

Tends to return a sparse point (sometimes, the sparsest)



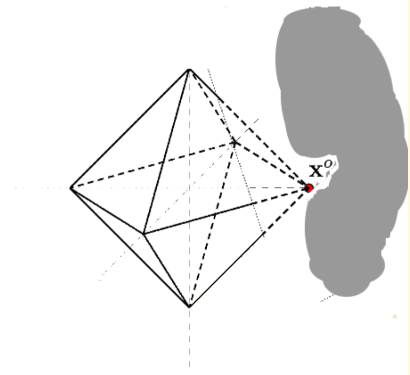
ℓ_1 ball touches the set

Example of sparse optimization

$$\min\{\|x\|_1 : \boxed{h(x) \le b}\}$$

Basis pursuit: find least 1-norm point on a **nonlinear set**

Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the set

LASSO denoises basis pursuit

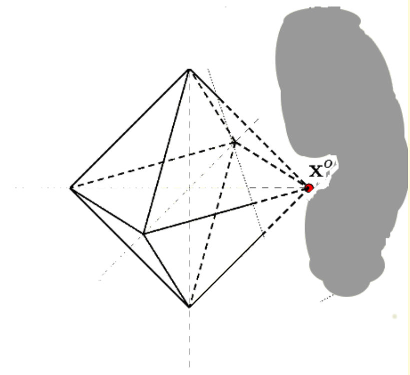
$$\min\left\{\|\boxed{h(x)} - b\|_2^2 : \|x\|_1 \le \tau\right\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : \boxed{h(x) \le b}\}$$

Basis pursuit: find least 1-norm point on a **nonlinear set**

Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the set

LASSO denoises basis pursuit

$$\min\left\{\|\boxed{h(x)} - b\|_2^2 : \|x\|_1 \le \tau\right\}$$

or

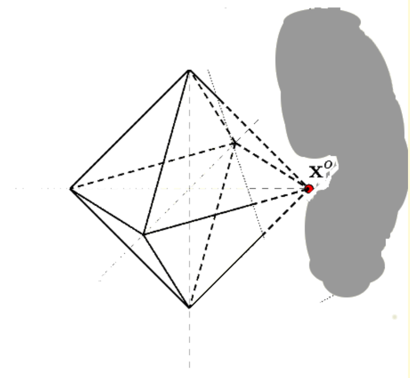
$$\min\left\{\|x\|_1 + \frac{\mu}{2}\|\boxed{h(x)} - b\|_2^2\right\}$$

Example of sparse optimization

$$\min\{\|x\|_1 : \boxed{h(x) \leq b}\}$$

Basis pursuit: find least 1-norm point on a **nonlinear set**

Tends to return a sparse point (sometimes, the sparsest)



ℓ_1 ball touches the set

LASSO denoises basis pursuit

$$\min\left\{\|\boxed{h(x)} - b\|_2^2 : \|x\|_1 \leq \tau\right\}$$

or

$$\min\left\{\|x\|_1 + \tfrac{\mu}{2}\|\boxed{h(x)} - b\|_2^2\right\}$$

or

$$\min\left\{\|x\|_1 : \|\boxed{h(x)} b\|_2^2 \leq \sigma\right\}$$

Lagrangian Relaxation Example

Real-life optimization problems

$$(\texttt{primal}) \quad \left\{ \begin{array}{ll} \min & \displaystyle\sum_{j\in J} \mathcal{C}^j(p^j) \\ & \text{for } j \in J\text{: } p^j \in \mathcal{P}^j \\ & \displaystyle\sum_{j\in J} g^j(p^j) = Dem \end{array} \right.$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \left\{
\begin{array}{ll}
\max & \sum_{j \in J} -\mathcal{C}^j(p^j) \\
& \text{for } j \in J\text{: } p^j \in \mathcal{P}^j \\
& \boxed{\sum_{j \in J} g^j(p^j) = Dem} \quad \boxed{\leftarrow x}
\end{array}
\right.
$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \left\{ \begin{array}{ll} \max & \displaystyle\sum_{j \in J} -\mathcal{C}^j(p^j) \\ & \text{for } j \in J\text{: } p^j \in \mathcal{P}^j \\ & \boxed{\displaystyle\sum_{j \in J} g^j(p^j) = Dem} \quad \boxed{\leftarrow x} \end{array} \right.
$$

have a (dual) with separable structure:

Lagrangian Relaxation Example

Real-life optimization problems

$$
\texttt{(primal)} \quad \begin{cases} \max & \sum_{j \in J} -\mathcal{C}^j(p^j) \\ & \text{for } j \in J\text{: } p^j \in \mathcal{P}^j \\ & \boxed{\sum_{j \in J} g^j(p^j) = Dem} \quad \leftarrow x \end{cases}
$$

have a `(dual)` with separable structure:

$$
\min_x \quad f(x) := \quad f_0(x) \quad + \sum_{j \in J} \quad f^j(x)
$$

$$
\min_x \quad -\langle x, Dem \rangle \quad + \sum_{j \in J} \begin{cases} \max & -\mathcal{C}^j(p^j) + \langle x, g^j(p^j) \rangle \\ & p^j \in \mathcal{P}^j \end{cases}
$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \left\{ \begin{array}{ll} \max & \displaystyle\sum_{j \in J} -\mathcal{C}^j(p^j) \\ & \text{for } j \in J \text{: } p^j \in \mathcal{P}^j \\ & \boxed{\displaystyle\sum_{j \in J} g^j(p^j) = Dem} \end{array} \right. \quad \boxed{\leftarrow x}
$$

have a (dual) with separable structure:

$$
\min_x \quad f(x) := \quad f_0(x) \quad + \sum_{j \in J} f^j(x)
$$

$$
\min_x \quad -\langle x, Dem\rangle \quad + \sum_{j \in J} \left\{ \begin{array}{ll} \max & -\mathcal{C}^j(p^j) + \langle x, g^j(p^j)\rangle \\ & p^j \in \mathcal{P}^j \end{array} \right.
$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
\texttt{(primal)} \quad \left\{ \begin{array}{ll} \max & \sum_{j \in J} -\mathcal{C}^j(p^j) \\ & \text{for } j \in J\text{: } p^j \in \mathcal{P}^j \\ & \boxed{\sum_{j \in J} g^j(p^j) = Dem} \end{array} \right. \quad \boxed{\leftarrow x}
$$

have a `(dual)` with separable structure:

$$
\min_x \quad f(x) := \quad f_0(x) \quad + \sum_{j \in J} \quad f^j(x)
$$

$$
\min_x \quad -\langle x, Dem \rangle \quad + \sum_{j \in J} \left\{ \begin{array}{ll} \max & -\mathcal{C}^j(p^j) + \langle x, g^j(p^j) \rangle \\ & p^j \in \mathcal{P}^j \end{array} \right.
$$

Similar for Benders Decomposition

Computing subgradients: how difficult is it?

1. $f(x) = |x|$, for $n = 1$
2. A linear Lasso function,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$
3. A nonlinear Lasso function, $h \in C^1$,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|\Big(h(x) - b\Big)^+\|_2^2$
4. One of the Lagrangian subproblems,
   $$f^j(x^k) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \langle x^k, g^j(p^j) \rangle \\ & p^j \in \mathcal{P}^j \end{cases}$$
5. The max-eigenvalue case
   $f(x^k) = \max\{y^\top A(x)y : \|y\|_2 \leq 1\}$

Computing subgradients: how difficult is it?

1. $f(x) = |x|$, for $n = 1$
2. A linear Lasso function,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$
3. A nonlinear Lasso function, $h \in C^1$,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|\Big(h(x) - b\Big)^+\|_2^2$
4. One of the Lagrangian subproblems,
   $$f^j(x^k) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \langle x^k, g^j(p^j) \rangle \\ & p^j \in \mathcal{P}^j \end{cases}$$
5. The max-eigenvalue case
   $f(x^k) = \max\{y^\top A(x)y : \|y\|_2 \leq 1\}$

What about computing the full $\partial f(x^k)$?

Computing subgradients: how difficult is it?

1. $f(x) = |x|$, for $n = 1$
2. A linear Lasso function,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_2^2$
3. A nonlinear Lasso function, $h \in C^1$,
   $f(x) = \|x\|_1 + \frac{\mu}{2}\|\Big(h(x) - b\Big)^+\|_2^2$
4. One of the Lagrangian subproblems,
   $$f^j(x^k) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \langle x^k, g^j(p^j)\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$$
5. The max-eigenvalue case
   $f(x^k) = \max\{y^\top A(x)y : \|y\|_2 \leq 1\}$

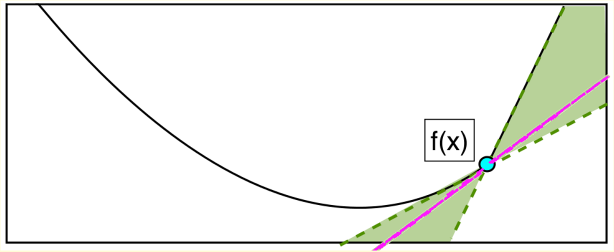
What about computing the full $\partial f(x^k)$?

NSO methods in general are designed

for oracles delivering 1-subgradient only

What can be done with the oracle output?

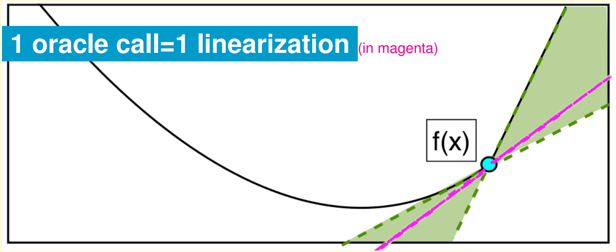
An example of a convex nonsmooth function



$$\begin{aligned}\partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top(y - x) \text{ for all y}\} \\ &= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}\end{aligned}$$

What can be done with the oracle output?

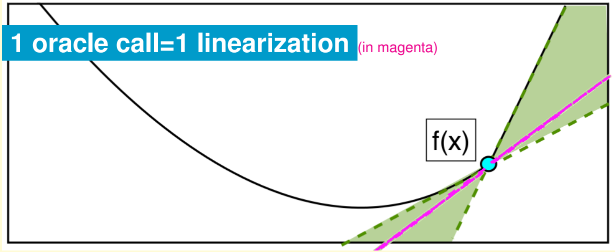
An example of a convex nonsmooth function



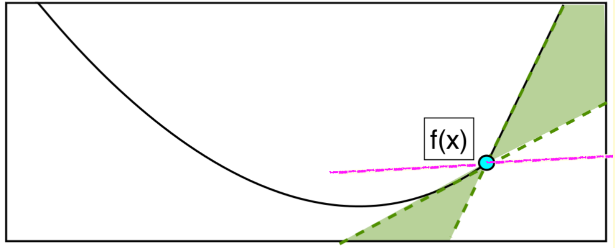
$$\begin{aligned}\partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all y}\} \\ &= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}\end{aligned}$$

What can be done with the oracle output?

An example of a convex nonsmooth function



Linearization bad if oracle output is bad
wrong *g* can give a bad linearization



Using oracle subgradients in the stopping test

Algorithms for unconstrained **smooth** optimization use as optimality certificate Fermat's rule

$$0 = \nabla f(\bar{x})$$

and generate a *minimizing sequence*:

$$\{x^k\} \to \bar{x} \text{ such that } \nabla f(x^k) \to 0\,.$$

If $f \in C^1$, then $\nabla f(\bar{x}) = 0$

Using oracle subgradients in the stopping test

Algorithms for unconstrained **smooth** optimization use as optimality certificate Fermat's rule

$$0 = \nabla f(\bar{x})$$

and generate a *minimizing sequence*:

$$\{x^k\} \to \bar{x} \text{ such that } \nabla f(x^k) \to 0\,.$$

If $f \in C^1$, then $\nabla f(\bar{x}) = 0$

In NSO things are less straightforward...

Using oracle subgradients in the stopping test

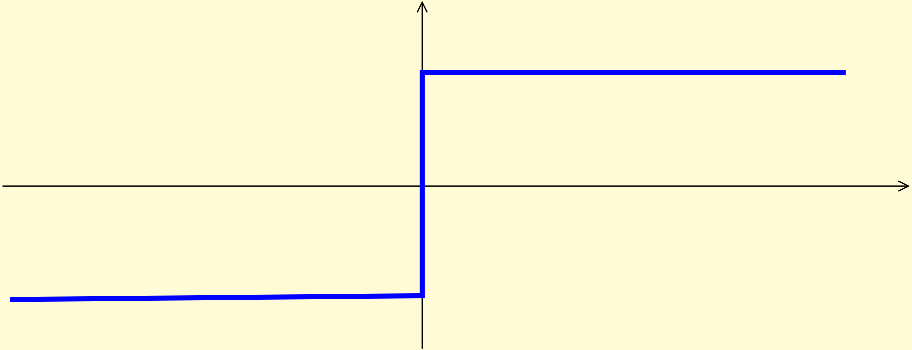
For the absolute value function, $f(x) = |x|$ and

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

Using as optimality certificate the inclusion

$$0 \in \partial f(\bar{x})$$

requires knowing the whole subdifferential!



NSO pitfalls

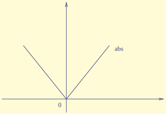
NSO pitfalls

NSO pitfalls

NSO pitfalls

Why special NSO methods?

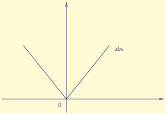
Smooth optimization techniques **do not work**



$$\begin{aligned} f(x) &= |x| \\ |\nabla f(x^k)| &= 1,\ \forall x^k \neq 0 \quad \partial f(0) = [-1, 1] \end{aligned}$$

Why special NSO methods?

Smooth optimization techniques **do not work**



$$\begin{aligned} f(x) &= |x| \\ |\nabla f(x^k)| &= 1,\ \forall x^k \neq 0 \quad \partial f(0) = [-1, 1] \end{aligned}$$

- Smooth stopping test **fails**:
  $|\nabla f(x^k)| \leq \text{TOL} \qquad (\leftrightarrow |g(x^k)| \leq \text{TOL})$

Why special NSO methods?

Smooth optimization techniques **do not work**

- Finite differences **fail**

  For $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x) = \max(x_1, x_2, x_3)$

  $\partial f(0) =?$

Why special NSO methods?

Smooth optimization techniques **do not work**

- Finite differences **fail**

  For $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x) = \max(x_1, x_2, x_3)$

  $\partial f(0) = ?$

Forward finite difference $\frac{f(x+\Delta x)-f(x)}{\Delta x}$

Why special NSO methods?

Smooth optimization techniques **do not work**

- Finite differences **fail**

  For $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x) = \max(x_1, x_2, x_3)$

  $\partial f(0) = ?$

Forward finite difference $\frac{f(x+\Delta x)-f(x)}{\Delta x}$

Central finite difference $\frac{f(x+\Delta x)-f(x-\Delta)}{2\Delta x}$

Why special NSO methods?

Smooth optimization techniques **do not work**

- Finite differences **fail**

  For $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x) = \max(x_1, x_2, x_3)$

  $\partial f(0) =$unit simplex

Forward finite difference $\frac{f(x+\Delta x)-f(x)}{\Delta x}$ $= (1, 1, 1)$

Central finite difference $\frac{f(x+\Delta x)-f(x-\Delta)}{2\Delta x}$ $= (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}$

Why special NSO methods?

Smooth optimization techniques **do not work**

- Finite differences **fail**

  For $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x) = \max(x_1, x_2, x_3)$

  $\partial f(0) =$unit simplex

Forward finite difference $\frac{f(x+\Delta x)-f(x)}{\Delta x}$ $= (1, 1, 1)$

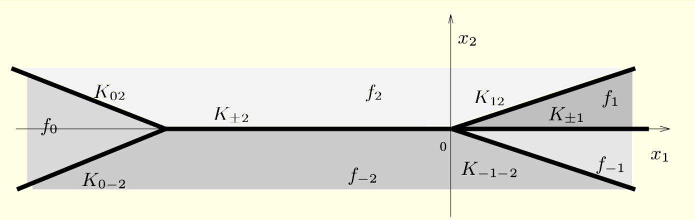
Central finite difference $\frac{f(x+\Delta x)-f(x-\Delta)}{2\Delta x}$ $= (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}$

none of them in the subdifferential!

Why special NSO methods?

Smooth optimization techniques **do not work**

- Linesearches get trapped in kinks and **fail**

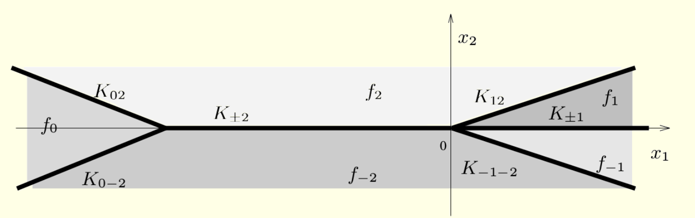


Example 9.1, “Instability of steepest descent”

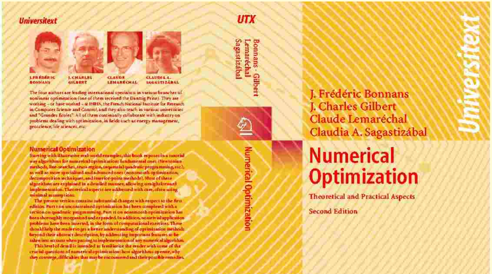
Why special NSO methods?

Smooth optimization techniques **do not work**

- Linesearches get trapped in kinks and **fail**



Example 9.1, "Instability of steepest descent"



Why special NSO methods?

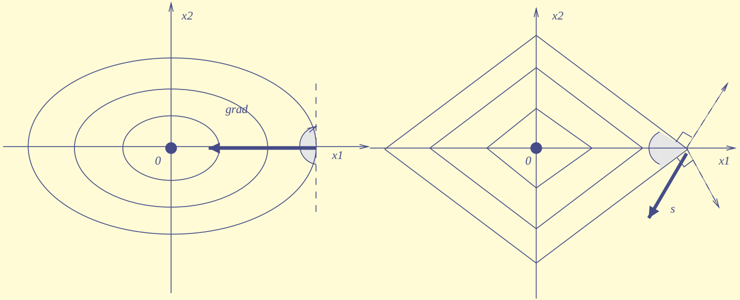
Smooth optimization techniques **do not work** (suite)

$-g(x^k)$ may **not** provide descent

Why special NSO methods?

Smooth optimization techniques **do not work** (suite)

$-g(x^k)$ may **not** provide descent



Direction opposite to a subgradient may **increase** the functional values

Why special NSO methods?

Smooth optimization techniques **do not work**

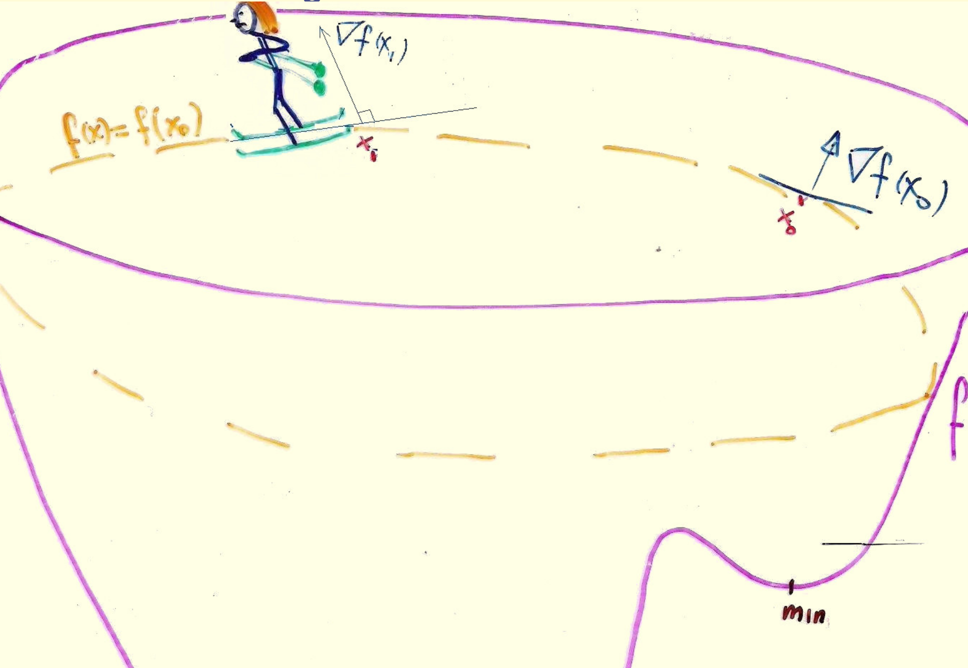
Smooth stopping test **fails**

Finite difference approximations **fail**

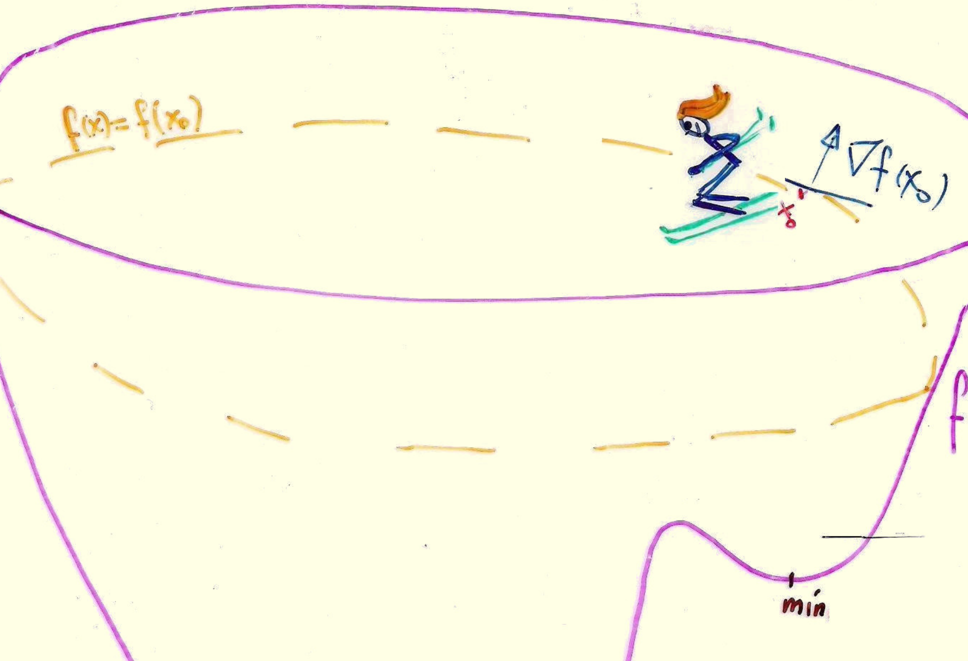
Linesearches get trapped in kinks and **fail**

Direction opposite to a subgradient may **increase** the functional values

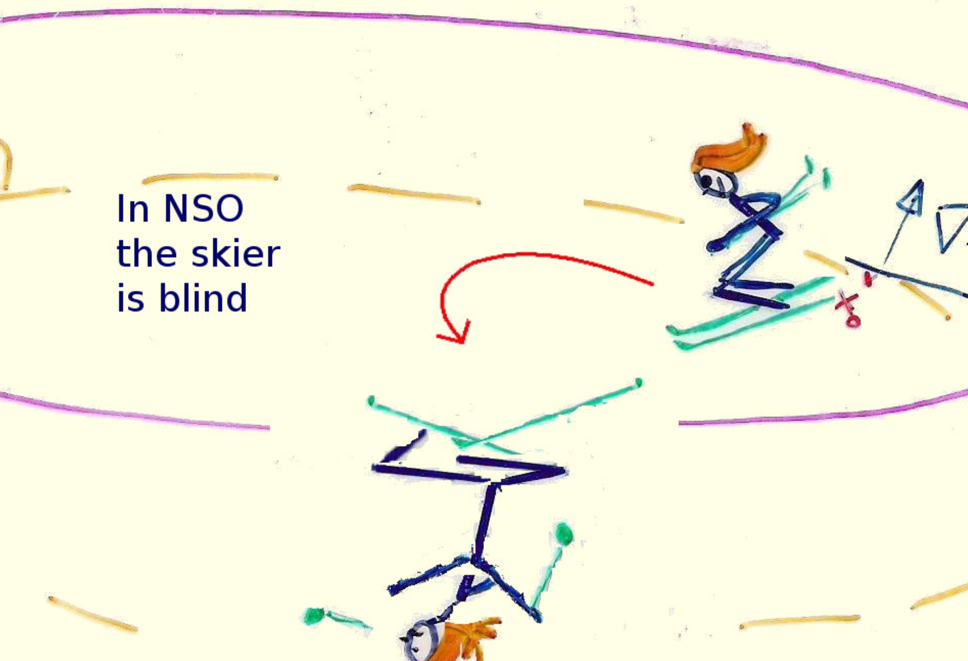
Nonsmooth skiing



Nonsmooth skiing



Nonsmooth skiing



Looking for sound optimality certificates in NSO

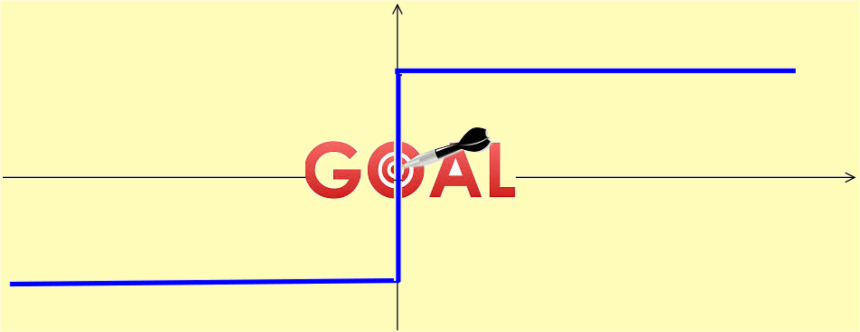
For the absolute value function, $f(x) = |x|$ and

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

Using as optimality certificate the inclusion

$$0 \in \partial f(\bar{x})$$

requires knowing the whole subdifferential!



Looking for sound optimality certificates in NSO

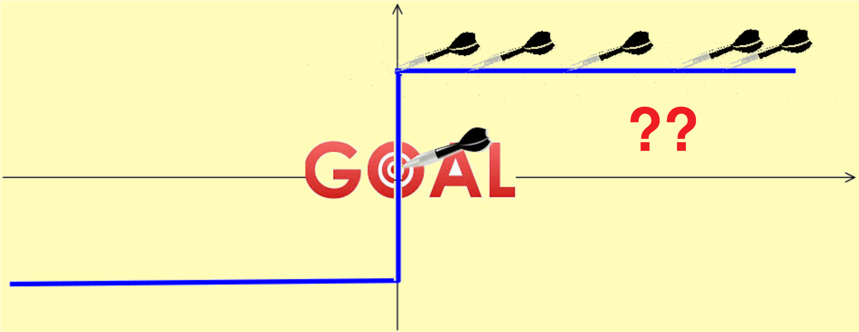
For the absolute value function, $f(x) = |x|$ and

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

Using as optimality certificate the inclusion

$$0 \in \partial f(\bar{x})$$

requires knowing the whole subdifferential!



What happens with the stopping test?

In **nonsmooth** optimization the inclusion

$$0 \in \partial f(\bar{x})$$

fails as optimality certificate

- As a set-valued mapping $\partial f(x)$ is osc:

$$\left(x^k, g(x^k) \in \partial f(x^k)\right) : \begin{cases} x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial f(\bar{x})$$

What happens with the stopping test?

In **nonsmooth** optimization the inclusion

$$0 \in \partial f(\bar{x})$$

fails as optimality certificate

- As a set-valued mapping $\partial f(x)$ is osc:

$$\left(x^k, g(x^k) \in \partial f(x^k)\right) : \begin{cases} x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial f(\bar{x})$$

- As a set-valued mapping, $\partial f(x)$ is **not** isc:
  Given $\bar{g} \in \partial f(\bar{x})$

$$\exists \left(x^k, g(x^k) \in \partial f(x^k)\right) : \begin{cases} x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases}$$

What happens with the stopping test?

In **nonsmooth** optimization the inclusion

$$0 \in \partial f(\bar{x})$$

fails as optimality certificate

- As a set-valued mapping $\partial f(x)$ is osc:

$$\left(x^k, g(x^k) \in \partial f(x^k)\right) : \begin{cases} x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial f(\bar{x})$$

- As a set-valued mapping, $\partial f(x)$ is **not** isc:
Given $\bar{g} \in \partial f(\bar{x})$

$$\nexists \left(x^k, g(x^k) \in \partial f(x^k)\right) : \begin{cases} x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases}$$

What happens with the stopping test?

We need to device a sound stopping test that does not rely on the straightforward extension of Fermat's rule

$$0 \in \partial f(\bar{x})$$

What happens with the stopping test?

We need to device a sound stopping test that does not rely on the straightforward extension of Fermat's rule

$$0 \in \partial f(\bar{x})$$

We use instead

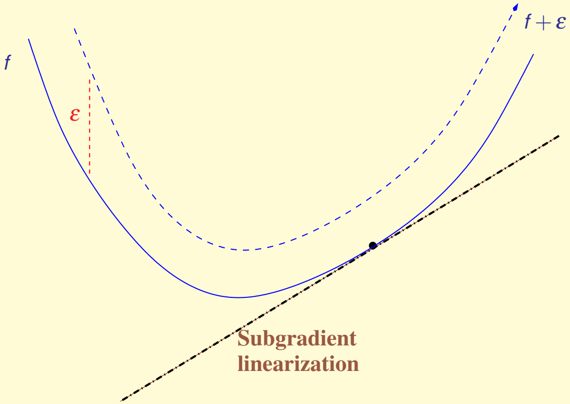
$$\bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x}) \quad \text{for } \|\bar{g}\| \text{ and } \bar{\varepsilon} \text{ small}$$

where the ε-subdifferential contains the slopes of linearizations supporting f **up to ε**, tangent at x:

$$\partial_\varepsilon f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) - \varepsilon \text{ for all y}\}$$

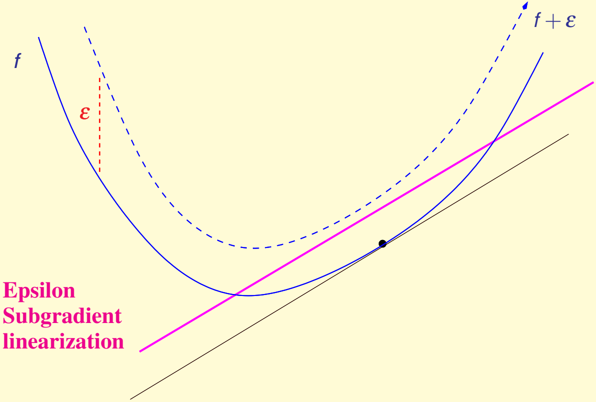
The ε-subdifferential

$$\partial_\varepsilon f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \boxed{-\varepsilon} \text{ for all y}\}$$



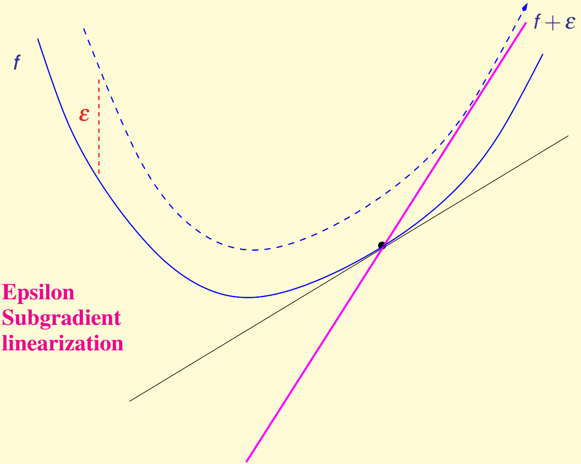
The ε-subdifferential

$$\partial_\varepsilon f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \boxed{-\varepsilon} \text{ for all y}\}$$



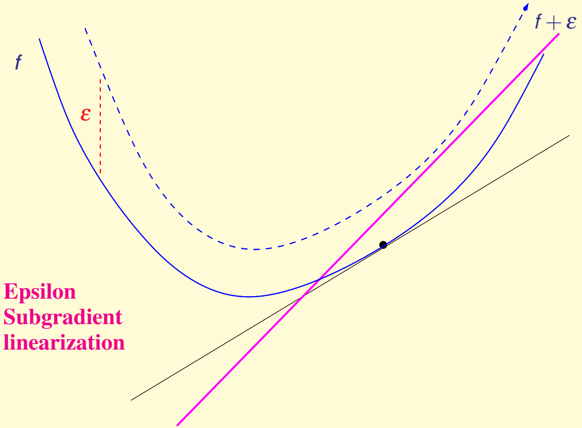
The ε-subdifferential

$$\partial_\varepsilon f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top(y-x) \boxed{-\varepsilon} \text{ for all y}\}$$



The ε-subdifferential

$$\partial_\varepsilon f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y-x) \boxed{-\varepsilon} \text{ for all y}\}$$

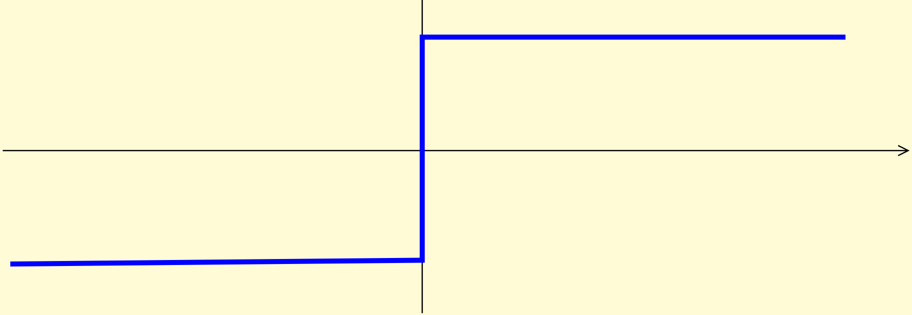


The ε-subdifferential

For the absolute value function, $f(x) = |x|$

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

$$\partial_\varepsilon f(x) = \begin{cases} [-1, -1 - \varepsilon/x] & x < -\varepsilon/2, \\ [-1, 1] & -\varepsilon/2 \le x1 \le \varepsilon 1/2, \\ [1 - \varepsilon/x, 1] & x > \varepsilon/2. \end{cases}$$

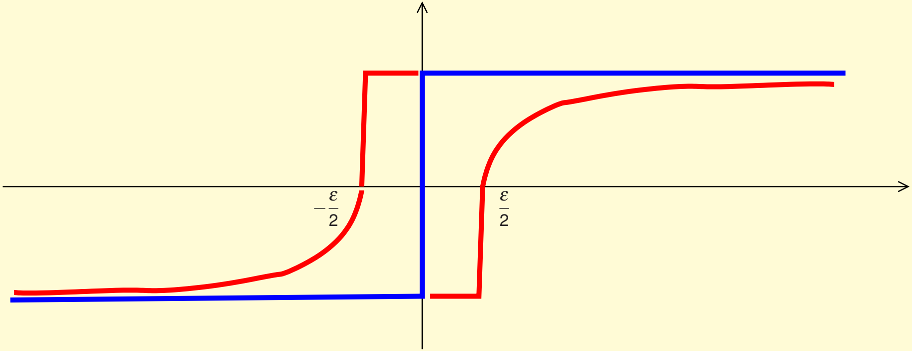


The ε-subdifferential

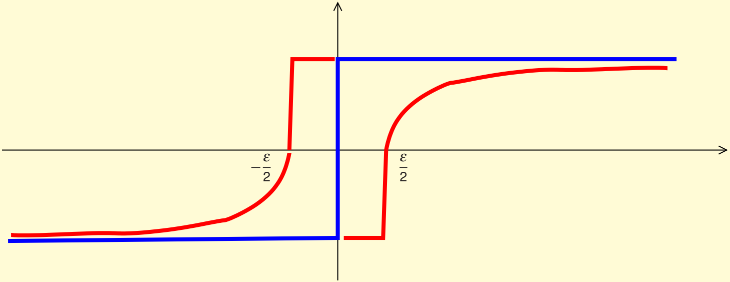
For the absolute value function, $f(x) = |x|$

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

$$\partial_\varepsilon f(x) = \begin{cases} [-1, -1-\varepsilon/x] & \text{if } x < -\varepsilon/2, \\ [-1, 1] & \text{if } -\varepsilon/2 \le x \le \varepsilon/2, \\ [1-\varepsilon/x, 1] & \text{if } x > \varepsilon/2. \end{cases}$$



The ε-subdifferential



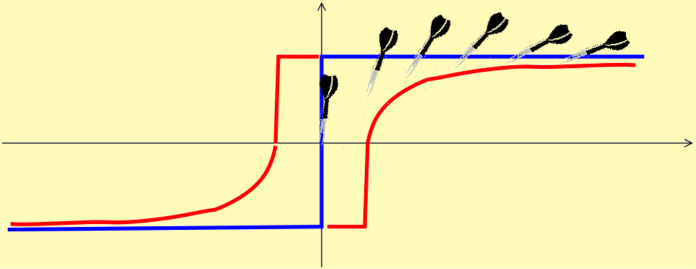
- As a set-valued mapping $\partial_\varepsilon f(x)$ is osc:

$$\left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \varepsilon \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \implies \bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$$

- As a set-valued mapping, $\partial_\varepsilon f(x)$ is isc: Given $\bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$

$$\exists \left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \bar{\varepsilon} \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases}$$

The ε-subdifferential



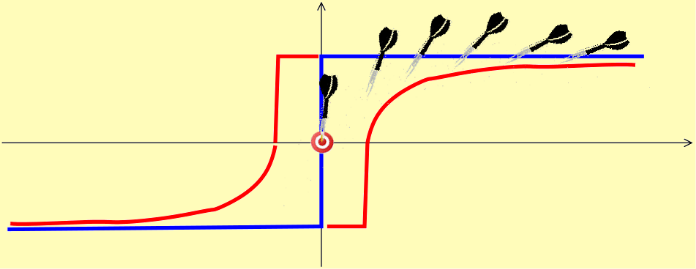
- As a set-valued mapping $\partial_\varepsilon f(x)$ is osc:

$$\left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \varepsilon \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$$

- As a set-valued mapping, $\partial_\varepsilon f(x)$ is isc: Given $\bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$

$$\exists\left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \bar{\varepsilon} \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases}$$

The ε-subdifferential



- As a set-valued mapping $\partial_\varepsilon f(x)$ is osc:

$$\left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \varepsilon \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$$

- As a set-valued mapping, $\partial_\varepsilon f(x)$ is isc: Given $\bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$

$$\exists \left(\varepsilon^k, x^k, g(x^k) \in \partial_{\varepsilon^k} f(x^k)\right) : \begin{cases} \varepsilon^k \to \bar{\varepsilon} \\ x^k \to \bar{x} \\ g(x^k) \to \bar{g}\,. \end{cases}$$

The ε-subdifferential and bundle methods

Generate iterates so that for a **subsequence $\{\hat{x}^k\}$**

- As a set-valued mapping $\partial_\varepsilon f(x)$ is osc:

$$\left(\hat{\varepsilon}^k, \hat{x}^k, \hat{g}(x^k) \in \partial_{\varepsilon^k} f(\hat{x}^k)\right) : \begin{cases} \hat{\varepsilon}^k \to \bar{\varepsilon} \\ x^k \to \bar{x} \\ \hat{g}(x^k) \to \bar{g}\,. \end{cases} \Longrightarrow \bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$$

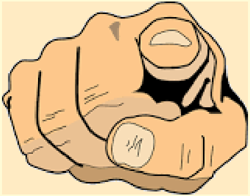
with $\bar{\varepsilon} \approx 0$ and $\|\bar{g}\| \approx 0$

- As a set-valued mapping, $\partial_\varepsilon f(x)$ is isc:
  Given $\bar{g} \in \partial_{\bar{\varepsilon}} f(\bar{x})$:

$$\exists \left(\hat{\varepsilon}^k, \hat{x}^k, \hat{g}(x^k) \in \partial_{\hat{\varepsilon}^k} f(\hat{x}^k)\right) : \begin{cases} \hat{\varepsilon}^k \to \bar{\varepsilon} \\ \hat{x}^k \to \bar{x} \\ \hat{g}(\hat{x}^k) \to \bar{g}\,. \end{cases}$$

Building up ε-subgradients in bundle methods

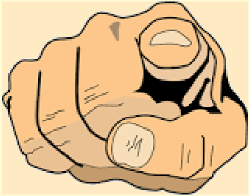
You told us



we were going to use subgradient information provided by a black-box because knowing the full subdifferential is too much of a requirement.

Building up ε-subgradients in bundle methods

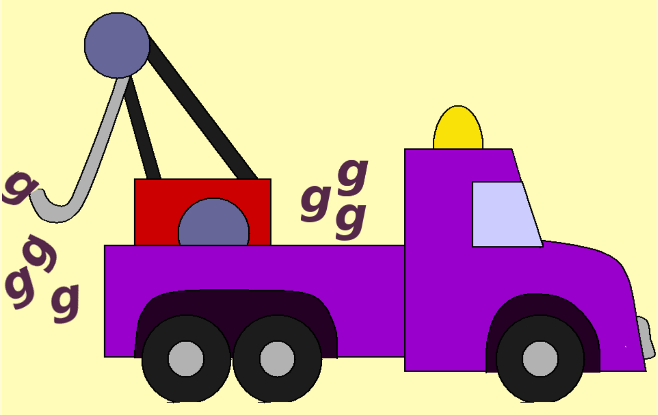
You told us



we were going to use subgradient information provided by a black-box because knowing the full subdifferential is too much of a requirement.

Now you want to use the ε-subdifferential, an even larger set!

The transportation formula



Or how to express subgradients at x^i as ε-subgradients at $\hat{x}^k$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$f(y) \quad \geq \quad f(x^i) + g^{i\top}(y - x^i)$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$
\begin{aligned}
f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\
&= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k)
\end{aligned}
$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$\begin{aligned} f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\ &= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k) \\ &= f(\hat{x}^k) + g^{i\top}(y - x^i) - \Big(f(\hat{x}^k) - f(x^i)\Big) \end{aligned}$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$
\begin{aligned}
f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\
&= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i \pm \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i)\Big)
\end{aligned}
$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$
\begin{aligned}
f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\
&= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i \pm \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i)\Big)
\end{aligned}
$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

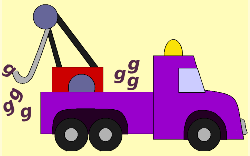
$$
\begin{aligned}
f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\
&= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i \pm \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - \hat{x}^k) - e^i(\hat{x}^k)
\end{aligned}
$$

The transportation formula

Consider $g^i \in \partial f(x^i)$

The inclusion holds if and only if, for all $y \in \mathbb{R}^n$

$$
\begin{aligned}
f(y) &\geq f(x^i) + g^{i\top}(y - x^i) \\
&= f(x^i) + g^{i\top}(y - x^i) \pm f(\hat{x}^k) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - x^i \pm \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - \hat{x}^k) - \Big(f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i)\Big) \\
&= f(\hat{x}^k) + g^{i\top}(y - \hat{x}^k) - e^i(\hat{x}^k)
\end{aligned}
$$



$$\Longrightarrow \boxed{g^i \in \partial_{e^i(\hat{x}^k)} f(\hat{x}^k)}$$

for $e^i(\hat{x}^k) := f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i) \geq 0$

The ε-subdifferential and bundle methods

We collect the black-box

 output at past iterations $x^i, i = 1, 2, \ldots, k$,

so that at iteration k we can define a **bundle** of information, centered at a special iterate $\hat{x}^k \in \{x^i\}$

$$\mathcal{B}^k := \begin{pmatrix} e^i(\hat{x}^k) = f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i) \\ g^i \in \partial_{e^i(\hat{x}^k)} f(\hat{x}^k) \end{pmatrix}$$

The ε-subdifferential and bundle methods

We collect the black-box

 output at past iterations $x^i, i = 1, 2, \ldots, k$,

so that at iteration k we can define a **bundle** of information, centered at a special iterate $\hat{x}^k \in \{x^i\}$

$$\mathcal{B}^k := \left(\begin{array}{c} e^i(\hat{x}^k) = f(\hat{x}^k) - f(x^i) - g^{i\top}(\hat{x}^k - x^i) \\ g^i \in \partial_{e^i(\hat{x}^k)} f(\hat{x}^k) \end{array} \right)$$

A suitable convex combination

$$\varepsilon^k := \sum_{i \in \mathcal{B}^k} \alpha^i e^i(\hat{x}^k) \text{ and } G^k := \sum_{i \in \mathcal{B}^k} \alpha^i g^i$$

will eventually satisfy the optimality condition!

Back to Computational NSO

For the unconstrained problem

$$\boxed{\min f(x)},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex but not differentiable at some points, we shall define algorithms based on information provided by an *oracle* or “black box”  endowed with reliable **stopping tests**

How is the oracle information used?

We look for algorithms based on information provided

by an *oracle* 

endowed with reliable stopping tests

How is the oracle information used?

We look for algorithms based on information provided

by an *oracle* 

~~endowed with reliable stopping tests~~

Subgradient methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

How is the oracle information used?

We look for algorithms based on information provided

by an *oracle* 

~~endowed with reliable stopping tests~~

Subgradient methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

Pros and cons of these methods?

How is the oracle information used?

We look for algorithms based on information provided

by an *oracle* 

~~endowed with reliable stopping tests~~

Subgradient methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

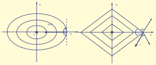
3 Make $k = k + 1$ and loop to 1.

Pros and cons of these methods?

Subgradient methods: pros and cons

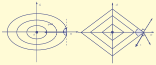
- Simple to code

Subgradient methods: pros and cons

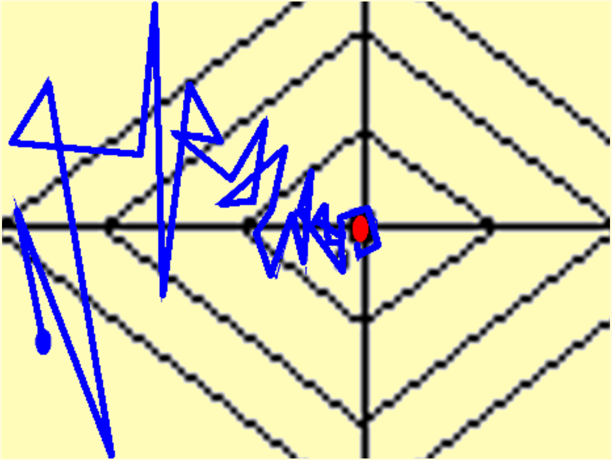


- Simple to code
- Generate non-monotone functional values

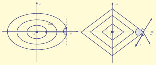
Subgradient methods: pros and cons



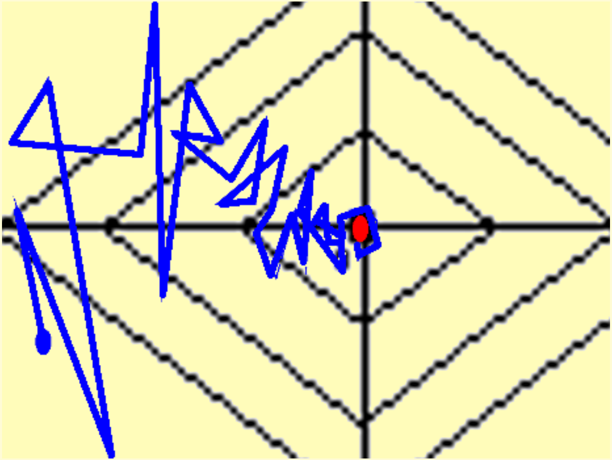
- Simple to code
- Generate non-monotone functional values



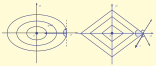
Subgradient methods: pros and cons



- Simple to code
- Generate non-monotone functional values

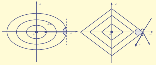


Subgradient methods



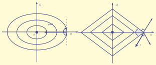
- Simple to code
- Generate non-monotone functional values
- Converge to a minimizer for $\{t_k\} \in \ell_2 \backslash \ell_\infty$ (eventually distance to solution set decreases)

Subgradient methods



- Simple to code
- Generate non-monotone functional values
- Converge to a minimizer for $\{t_k\} \in \ell_2 \backslash \ell_\infty$ (eventually distance to solution set decreases)
- **Lacks a stopping test**

Subgradient methods



- Simple to code
- Generate non-monotone functional values
- Converge to a minimizer for $\{t_k\} \in \ell_2 \backslash \ell_\infty$ (eventually distance to solution set decreases)
- **Lacks a stopping test**

…**does not use all available information!**

f(x)??

$g(x) \in \partial f(x)$

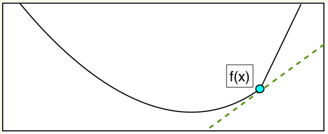
How is the oracle information used?

We look for algorithms based on information provided by an *oracle*  endowed with reliable stopping tests

How is the oracle information used?

We look for algorithms based on information provided by an *oracle*  endowed with reliable stopping tests

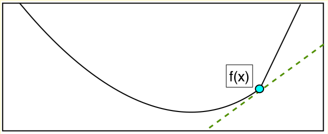
Black box information defines linearizations



How is the oracle information used?

We look for algorithms based on information provided by an *oracle*  endowed with reliable stopping tests

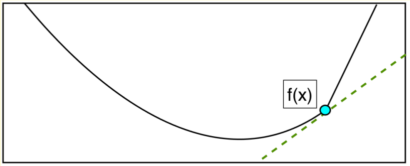
Black box information defines linearizations



that put together create a **model M** of the function f.

The model is used to define iterates and to put in place a stopping test

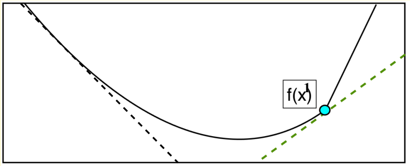
How is the oracle information used?



Black box information defines linearizations that put together create a **model M** of the function f.

x^i  $\begin{array}{l} f^i = f(x^i) \\ g^i = g(x^i) \end{array} \Rightarrow$ $f^i + g^{i\top}(x - x^i)$

How is the oracle information used?

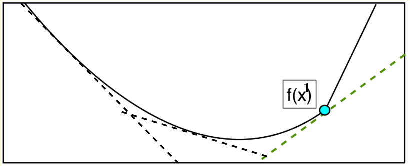


Black box information defines linearizations that put together create a **model M** of the function f.

x^i 

$$\begin{array}{l} f^i = f(x^i) \\ g^i = g(x^i) \end{array} \Rightarrow \mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$$

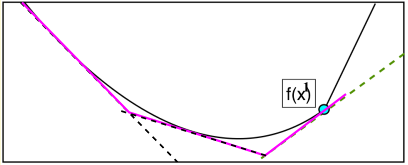
How is the oracle information used?



Black box information defines linearizations that put together create a **model M** of the function f.

x^i  $\begin{array}{l} f^i = f(x^i) \\ g^i = g(x^i) \end{array} \Rightarrow \mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

How is the oracle information used?



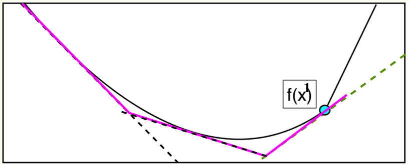
Black box information defines linearizations that put together create a **model M** of the function f.

x^i 

$$f^i = f(x^i)$$
$$g^i = g(x^i)$$

$$\Rightarrow \mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$$

How is the oracle information used?



Black box information defines linearizations that put together create a **model M** of the function f.

x^i 

$$\begin{array}{l} f^i = f(x^i) \\ g^i = g(x^i) \end{array} \Rightarrow \mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$$

For future use: $\partial \mathbf{M}(x) = conv\{g^i : i \in I(x)\}$

Cutting-plane methods

To minimize f (unavailable in an explicit manner),

minimize its model $\boxed{\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}}$

Improve the model at each iteration

Cutting-plane methods

To minimize f (unavailable in an explicit manner),
minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$
Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_k(x) &= \max_{i \le k} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\Big(\mathbf{M}_{k-1}(x), f^k + g^{k\top}(x - x^k) \Big) \\ &\quad \textbf{where } x^k \textbf{ minimizes } \mathbf{M}_{k-1} \end{aligned}$$

Cutting-plane methods

To minimize f (unavailable in an explicit manner),
minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$
Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_k(x) &= \max_{i \le k} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\Big(\mathbf{M}_{k-1}(x), f^k + g^{k\top}(x - x^k) \Big) \\ & \textbf{ where } x^k \textbf{ minimizes } \mathbf{M}_{k-1} \end{aligned}$$

Instead of $x^* \in \arg\min f(x)$ **at one shot**

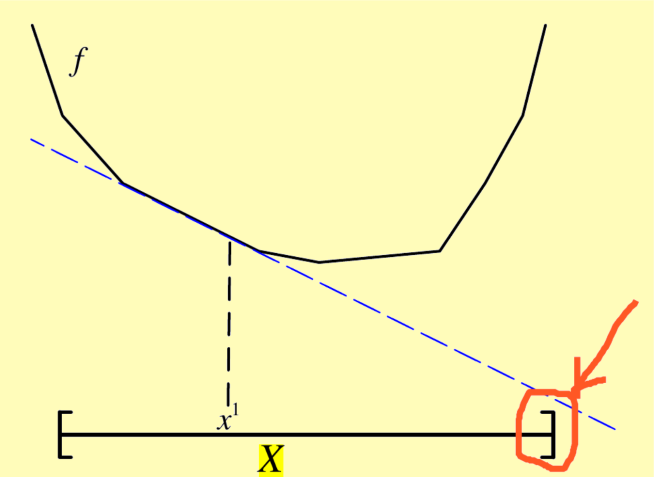
Cutting-plane methods

To minimize f (unavailable in an explicit manner),
minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$
Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_k(x) &= \max_{i \le k} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\Big(\mathbf{M}_{k-1}(x), f^k + g^{k\top}(x - x^k) \Big) \\ &\quad \textbf{where } x^k \textbf{ minimizes } \mathbf{M}_{k-1} \end{aligned}$$

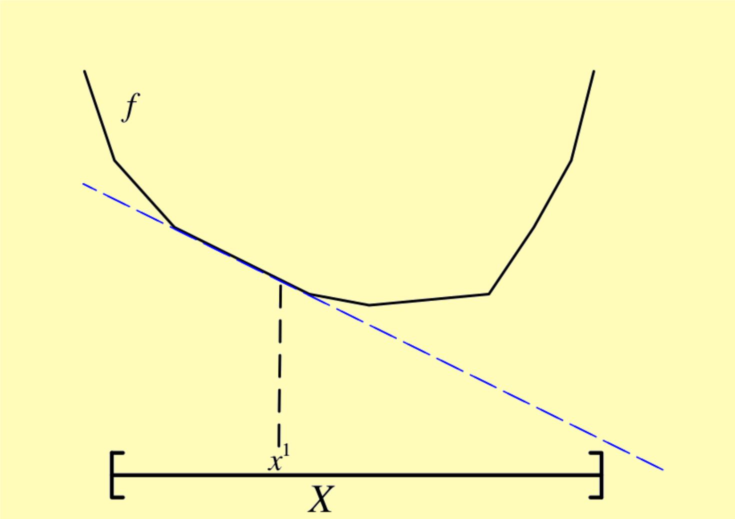
Instead of $x^* \in \arg\min f(x)$ **at one shot**
$x^k \in \arg\min \mathbf{M}_{k-1}(x)$ **iteratively**

Cutting-plane methods

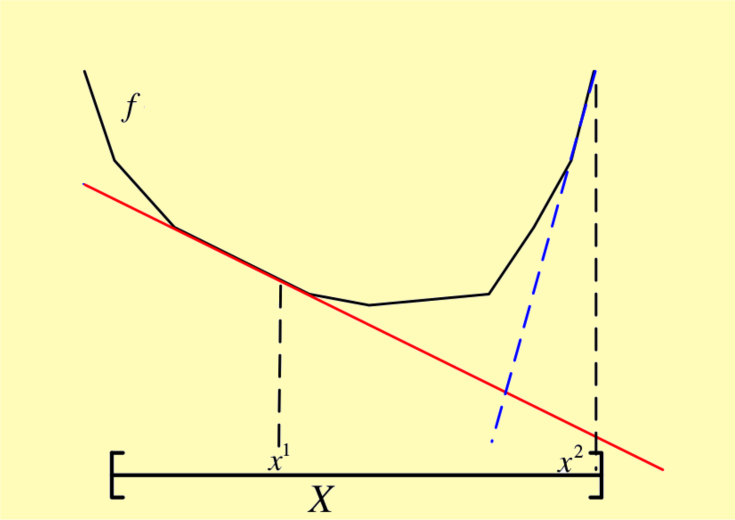


Artificial bounding at least for the first iterations

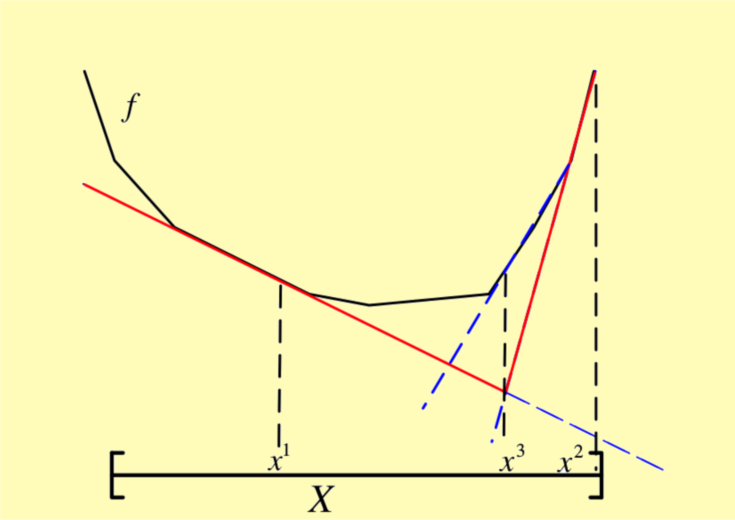
Cutting-plane methods



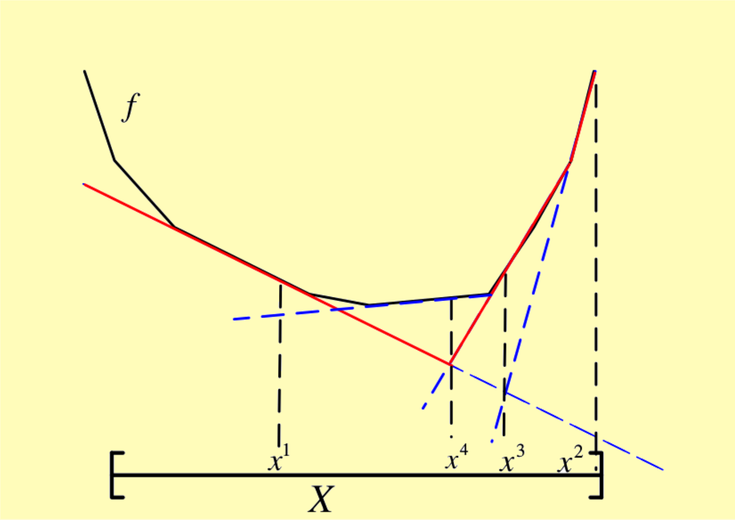
Cutting-plane methods



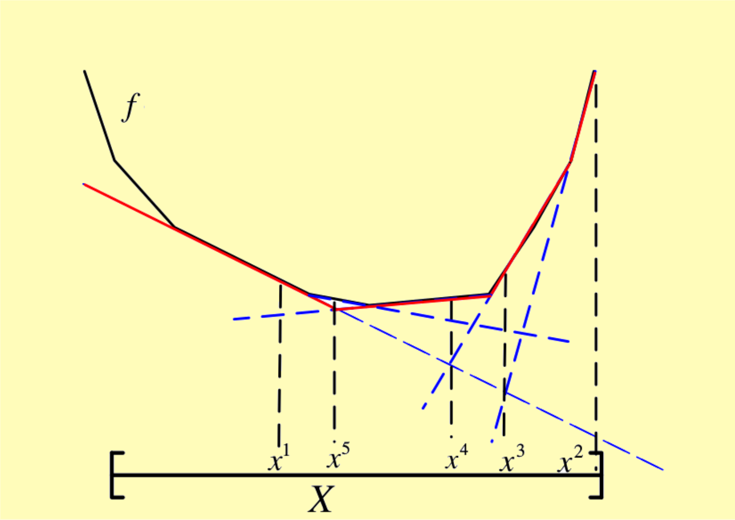
Cutting-plane methods



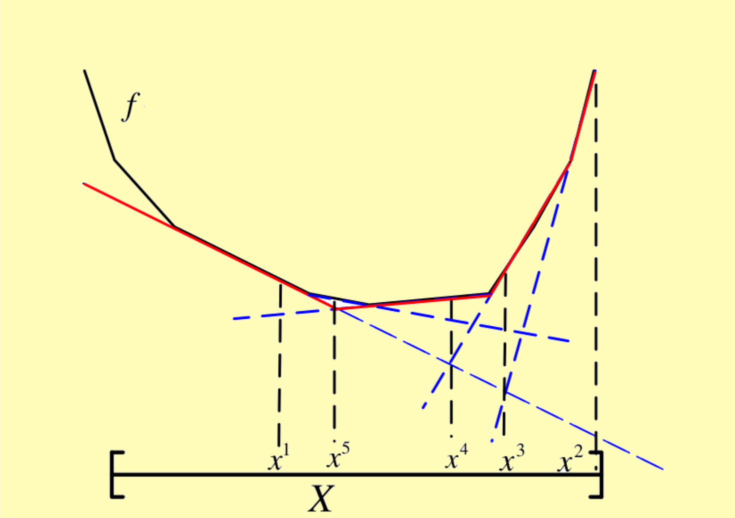
Cutting-plane methods



Cutting-plane methods

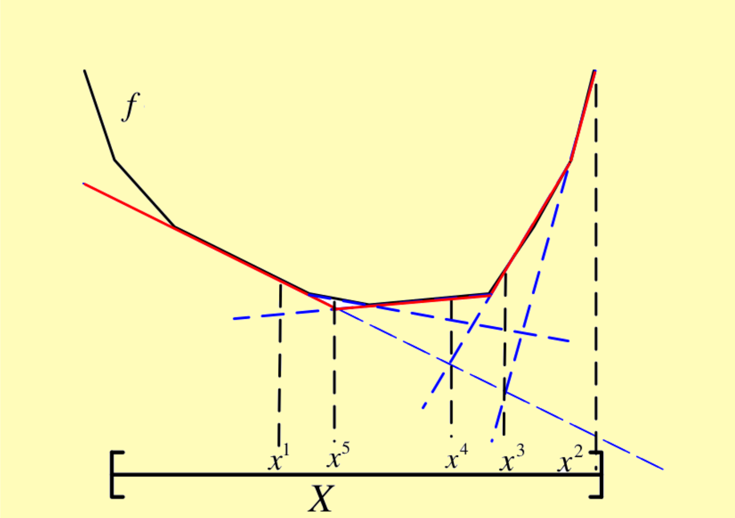


Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases

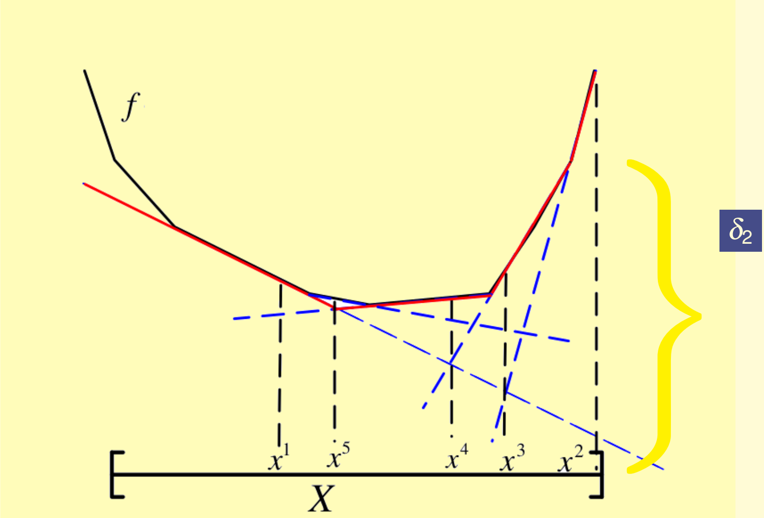
Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases, but functional values may not decrease

$f(x^2) > f(x^1)$

Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases, but functional values may not decrease

$f(x^2) > f(x^1)$

Optimality certificate checks if $\delta_k := f(x^k) - \mathbf{M}_{k-1}(x^k)$ is small

Cutting-plane methods

0 Choose x^1 and set $k = 1$ and $\mathbf{M}_0 \equiv -\infty$

1 Call the oracle at x^k.

$\delta_k = f(x^k) - \mathbf{M}_{k-1}(x^k) \leq tol$

2 Given $\mathbf{M}_k(\cdot) = \max\Big(\mathbf{M}_{k-1}(\cdot), f^k + g^{k\top}(\cdot - x^k)\Big)$

compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 Set $k = k + 1$, loop to 1.

Cutting-plane methods

0 Choose x^1 and set $k = 1$ and $\mathbf{M}_0 \equiv -\infty$

1 Call the oracle at x^k.
If $\delta_k = f(x^k) - \mathbf{M}_{k-1}(x^k) \leq tol$ STOP

2 Given $\mathbf{M}_k(\cdot) = \max\left(\mathbf{M}_{k-1}(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$

compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 Set $k = k + 1$, loop to 1.

Cutting-plane methods

0 Choose x^1 and set $k = 1$ and $\mathbf{M}_0 \equiv -\infty$

1 Call the oracle at x^k.
If $\delta_k = f(x^k) - \mathbf{M}_{k-1}(x^k) \leq tol$ STOP

2 Given $\mathbf{M}_k(\cdot) = \max\left(\mathbf{M}_{k-1}(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$

compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 Set $k = k + 1$, loop to 1.

In 2, $\mathbf{M}_k(x) = \max_{i \leq k}\{f^i + g^{i\top}(x - x^i)\}$ and X polyhedral.
$\Longrightarrow$ 2 $\equiv$ to solving a linear programming problem

$$\begin{cases} \min & r \\ \text{s.t.} & r \in \mathbb{R}, x \in X \\ & r \geq f^i + g^{i\top}(x - x^i) \text{ for } i \leq k \end{cases}$$

Cutting-plane methods: pros and cons

- One LP solve per iteration
- Require a sound choice of initial bounding set (polyhedral X)
- Generate non-monotone functional values

Cutting-plane methods: pros and cons

- One LP solve per iteration
- Require a sound choice of initial bounding set (polyhedral X)
- Generate non-monotone functional values
- Converge for a subsequence, as $\liminf \left(f(x^k) - \mathbf{M}_{k-1}(x^k) \right) \to 0$
- Have a good stopping test

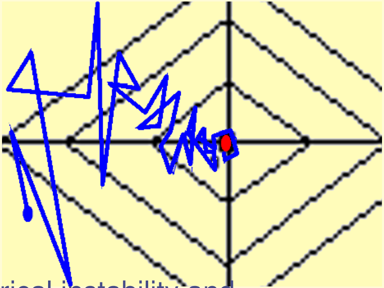
Cutting-plane methods: pros and cons

- One LP solve per iteration
- Require a sound choice of initial bounding set (polyhedral X)
- Generate non-monotone functional values
- Converge for a subsequence, as $\liminf\left(f(x^k) - \mathbf{M}_{k-1}(x^k)\right) \to 0$
- Have a good stopping test
- LP problem has more and more constraints, and eventually numerical errors prevail

$$\left\{\begin{array}{ll} \min & r \\ \text{s.t.} & r \in \mathbb{R}, x \in X \\ & r \geq f^i + g^{i\top}(x - x^i) \text{ for } i \leq \mathbf{k} \end{array}\right.$$

Cutting-plane methods: pros and cons

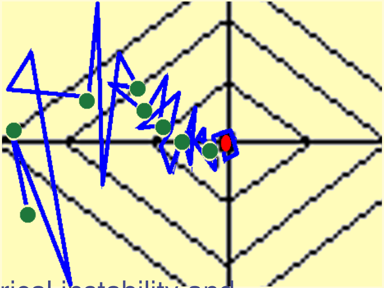
- CP methods bring in the concept of a model, which gives a stopping test
  (a subsequence of $\{\delta_k\} \to 0$)
- CP methods still non-monotone



Monotonicity is the key to defeat numerical instability and oscillations

Cutting-plane methods: pros and cons

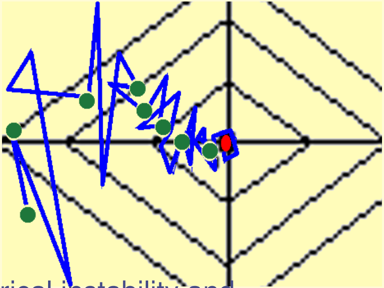
- CP methods bring in the concept of a model, which gives a stopping test
  (a subsequence of $\{\delta_k\} \to 0$)
- CP methods still non-monotone



Monotonicity is the key to defeat numerical instability and oscillations

Cutting-plane methods: pros and cons

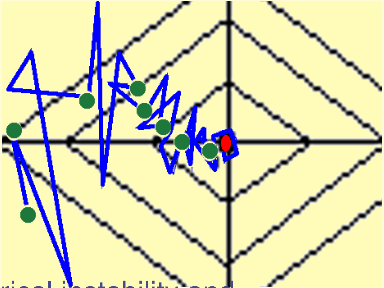
- CP methods bring in the concept of a model, which gives a stopping test
  (a subsequence of $\{\delta_k\} \to 0$)
- CP methods still non-monotone



Monotonicity is the key to defeat numerical instability and oscillations: the subsequence of functional values at green-spot iterates converges.

Cutting-plane methods: pros and cons

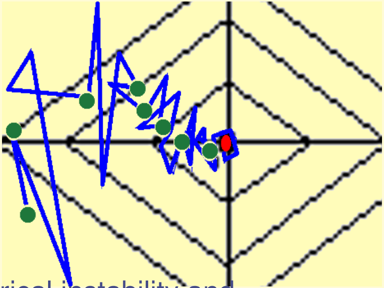
- CP methods bring in the concept of a model, which gives a stopping test
  (a subsequence of $\{\delta_k\} \to 0$)
- CP methods still non-monotone



Monotonicity is the key to defeat numerical instability and oscillations: the subsequence of functional values at green-spot iterates converges. Given $m \in (0, 1)$, the **descent rule** in bundle methods selects green-spots as follows:

Cutting-plane methods: pros and cons

- CP methods bring in the concept of a model, which gives a stopping test
  (a subsequence of $\{\delta_k\} \to 0$)
- CP methods still non-monotone



Monotonicity is the key to defeat numerical instability and oscillations: the subsequence of functional values at green-spot iterates converges. Given $m \in (0,1)$, the **descent rule** in bundle methods selects green-spots as follows:

$$f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k \Longrightarrow \hat{\mathbf{x}}^{\mathbf{k+1}} := \mathbf{x}^{\mathbf{k+1}}$$

Limit points of the **serious-step** subsequence $\{\hat{\mathbf{x}}^{\mathbf{k}}\}$ minimize f

Bundle methods

0 Choose x^1, $t_1 > 0$, and set $\hat{x}^1 = x^1$, $k = 1$.

1 Given $\hat{x}^k$, $\mathbf{M}_k$ and t_k, compute x^{k+1} and $\delta_{k+1} := f(\hat{x}^k) - \mathbf{M}_k(x^{k+1})$

2 Call the oracle at x^{k+1}. If $\delta_{k+1} \leq tol$ **STOP**

3 `(Descent Rule)`

$$f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k? \quad \begin{cases} \text{yes} & \text{SS: } \hat{x}^{k+1} = x^{k+1} \\ \text{no} & \text{NS: } \hat{x}^{k+1} = \hat{x}^k \end{cases}$$

4 Choose a new model and stepsize.

5 Set $k = k + 1$, loop to 1.

Bundle methods: the power of the descent rule

The serious-step subsequence satisfies the descent rule

$$f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k \qquad (\texttt{DR})$$

Rearranging terms, for k serious,

$$m\delta_k \leq f(\hat{x}^k) - f(\hat{x}^{k+1})$$

Bundle methods: the power of the descent rule

The serious-step subsequence satisfies the descent rule

$$f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k \qquad (\mathtt{DR})$$

Rearranging terms, for k serious,

$$m\delta_k \leq f(\hat{x}^k) - f(\hat{x}^{k+1})$$

(telescopic) sum yields

$$(0 \leq) \qquad m \sum_{k_{\text{serious}}} \delta_k \leq f(\hat{x}^0) - f^\infty \quad \text{for } f^\infty := \liminf f(\hat{x}^{k+1})$$

Bundle methods: the power of the descent rule

The serious-step subsequence satisfies the descent rule

$$f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k \qquad (\texttt{DR})$$

Rearranging terms, for k serious,

$$m\delta_k \leq f(\hat{x}^k) - f(\hat{x}^{k+1})$$

(telescopic) sum yields

$$(0 \leq) \qquad m \sum_{k_{\text{serious}}} \delta_k \leq f(\hat{x}^0) - f^\infty \quad \text{for } f^\infty := \liminf f(\hat{x}^{k+1})$$

As long as $\delta_k \geq 0$ in `(DR)`,

- Either $f^\infty = -\infty$ (problem unbounded below)
- Or $f^\infty > -\infty$, in which case

Bundle methods: the power of the descent rule

The serious-step subsequence satisfies the descent rule

$$f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k \qquad (\mathtt{DR})$$

Rearranging terms, for k serious,

$$m\delta_k \leq f(\hat{x}^k) - f(\hat{x}^{k+1})$$

(telescopic) sum yields

$$(0 \leq) \qquad m \sum_{k_{\text{serious}}} \delta_k \leq f(\hat{x}^0) - f^\infty \quad \text{for } f^\infty := \liminf f(\hat{x}^{k+1})$$

As long as $\delta_k \geq 0$ in (DR),

- Either $f^\infty = -\infty$ (problem unbounded below)
- Or $f^\infty > -\infty$, in which case
  - $\delta_k \to 0$

Bundle methods: what about non-serious points?

Without even specifying how iterates x^k are computed, if the descent rule holds for infinitely many iterates, the subsequence of serious optimality certificates (δ_k) goes to zero (and $\{\hat{x}^k\}$ will be minimizing).

Bundle methods: what about non-serious points?

Without even specifying how iterates x^k are computed, if the descent rule holds for infinitely many iterates, the subsequence of serious optimality certificates (δ_k) goes to zero (and $\{\hat{x}^k\}$ will be minimizing). The above ensures convergence for the serious subsequence

Bundle methods: what about non-serious points?

Without even specifying how iterates x^k are computed, if the descent rule holds for infinitely many iterates, the subsequence of serious optimality certificates (δ_k) goes to zero (and $\{\hat{x}^k\}$ will be minimizing). The above ensures convergence for the serious subsequence

What about iterates that do not satisfy `(DR)`?

These are the null steps in the bundle jargon.
Their role is to enrich the model and gain more information on f, so that eventually an iterate is accepted as a serious one.

Bundle methods: what about non-serious points?

Without even specifying how iterates x^k are computed, if the descent rule holds for infinitely many iterates, the subsequence of serious optimality certificates (δ_k) goes to zero (and $\{\hat{x}^k\}$ will be minimizing). The above ensures convergence for the serious subsequence

What about iterates that do not satisfy `(DR)`?

These are the null steps in the bundle jargon.
Their role is to enrich the model and gain more information on f, so that eventually an iterate is accepted as a serious one.

The analysis of the situation when there is a last serious iterate, $\hat{x}$, followed by infinitely many null steps, is related to the proximal point operator

A question for you:

Ever heard of the proximal point mapping?

Answer provided by Jean-Jacques Moreau

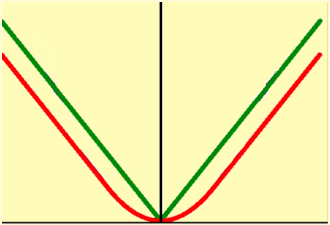
The Moreau-envelope of f is a $C^{1,1}$-smoothing of f

$$F_\mu(x) := \min\left\{ f(y) + \frac{1}{2}\mu \|y - x\|^2 \right\}$$

Answer provided by Jean-Jacques Moreau

The Moreau-envelope of f is a $C^{1,1}$-smoothing of f

$$F_\mu(x) := \min\left\{ f(y) + \frac{1}{2}\mu \|y - x\|^2 \right\}$$



- the unique minimizer is the proximal point mapping $p^f_\mu(x)$

- the envelope's gradient is $\nabla F_\mu(x) = \mu\Big(x - p\mu(x)\Big)$

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\Longleftrightarrow \quad \in \partial f(p) + \frac{1}{t}(p - x) \\
&\Longleftrightarrow
\end{aligned}
$$

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\iff \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\iff \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\
&\iff
\end{aligned}
$$

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\iff \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\iff \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\
&\iff \quad \boxed{\frac{1}{t}(x - p) \in \partial f(p)}
\end{aligned}
$$

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\Longleftrightarrow \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\
&\Longleftrightarrow \quad \boxed{\frac{1}{t}(x - p) \in \partial f(p)}
\end{aligned}
$$

The implicit inclusion cannot be solved without full knowledge of the subdifferential,

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\Longleftrightarrow \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\
&\Longleftrightarrow \quad \boxed{\frac{1}{t}(x - p) \in \partial f(p)}
\end{aligned}
$$

The implicit inclusion cannot be solved without full knowledge of the subdifferential, needs an informative oracle

How to compute the prox?

$$\begin{aligned} p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\ &\Longleftrightarrow \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\ &\Longleftrightarrow \quad \boxed{\frac{1}{t}(x - p) \in \partial f(p)} \end{aligned}$$

The implicit inclusion cannot be solved without full knowledge of the subdifferential, needs an informative oracle

note 1: $\Longleftrightarrow \; p \in x - t\partial f(p)$ **akin to a subgradient method**

How to compute the prox?

$$
\begin{aligned}
p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\Longleftrightarrow \quad 0 \in \partial f(p) + \frac{1}{t}(p - x) \\
&\Longleftrightarrow \quad \boxed{\frac{1}{t}(x - p) \in \partial f(p)}
\end{aligned}
$$

The implicit inclusion cannot be solved without full knowledge of the subdifferential, needs an informative oracle

note 1: $\Longleftrightarrow \; p \in x - t\partial f(p)$ **akin to a subgradient method**

note 2: $\bar{x}$ **minimizes** $f \; \Longleftrightarrow \; p = \bar{x} \; \Longleftrightarrow \; 0 \in \partial f(p) = \partial f(\bar{x})$

Proximal point algorithm (PPA) (Accel. Nesterov, FISTA)

Given starting point x^1 and prox-stepsize $t_1 0$,

$$x^{k+1} = p^f_{t_k}(x^k)$$

$$\iff$$

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

Proximal point algorithm (PPA) (Accel. Nesterov, FISTA)

Given starting point x^1 and prox-stepsize $t_1 0$,

$$x^{k+1} = p^f_{t_k}(x^k)$$

$$\Longleftrightarrow$$

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

- of interest only if computing $p^f_{t_k}(x^k)$ is much easier than minimizing f
- stepsize $t_k > 0$ impacts on the number of iterations (speed)

Proximal point algorithm (PPA) (Accel. Nesterov, FISTA)

Given starting point x^1 and prox-stepsize $t_1 0$,

$$x^{k+1} = p^f_{t_k}(x^k)$$
$$\Longleftrightarrow$$
$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

- of interest only if computing $p^f_{t_k}(x^k)$ is much easier than minimizing f
- stepsize $t_k > 0$ impacts on the number of iterations (speed)
- optimality certificate $\delta_k := \dfrac{x^{k+1} - x^k}{t_k} = \dfrac{p^f_{t_k}(x^k) - x^k}{t_k}$

Proximal point: calculus rules

- separable sum:
  $f(x, y) = g(x) + h(y) \Longrightarrow$
  $p_t^f(x) = \Big(p_t^g(x), p_t^h(y) \Big)$
- scalar factor ($\alpha \neq 0$) and translation ($v \neq 0$):
  $f(x) = g(\alpha x + v) \Longrightarrow$
  $p_t^f(x) = \frac{1}{\alpha} \Big(p_t^{\alpha^2 g}(\alpha x + v) - v \Big)$
- "perspective" ($\alpha > 0$):
  $f(x) = \alpha g(\frac{1}{\alpha} x) \Longrightarrow p_t^f(x) = \alpha p_t^{g/\alpha}(\frac{x}{\alpha})$

Proximal point: special functions

- + linear term ($v \neq 0$):
  $f(x) = g(x) + \langle v, x \rangle \Longrightarrow p_t^f(x) = p_t^g(x - v)$
- + convex quadratic term ($t > 0$):
  $f(x) = g(x) + \dfrac{1}{2t}\|x - v\|^2 \Longrightarrow$
  $p_t^f(x) = p_t^{\lambda g}(\lambda x + (1 - \lambda)v)$ for $\lambda = \dfrac{t}{t+1}$
- composition with linear term such that $A^\top A = \frac{1}{\alpha} I$, ($\alpha \neq 0$):
  $f(x) = g(Ax + v) \Longrightarrow$
  $p_t^f(x) = (I - \alpha A^\top A)x + \alpha A^\top \left[p_t^{g/\alpha}(Ax + v) - v \right]$

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

$$f(x^{k+1}) \le f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

(sharper inequality without $\frac{1}{2}$ also holds)

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

(sharper inequality without $\frac{1}{2}$ also holds)

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

- If $\arg\min f \neq \emptyset$ then $\delta_k = \|x^{k+1} - x^k\|^2 / t_k \to 0$

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

(sharper inequality without $\frac{1}{2}$ also holds)

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

- If $\arg\min f \neq \emptyset$ then $\delta_k = \|x^{k+1} - x^k\|^2 / t_k \to 0$
- If $\sum_k t_k \to \infty$ the sequence is minimizing

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

(sharper inequality without $\frac{1}{2}$ also holds)

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

- If $\arg\min f \neq \emptyset$ then $\delta_k = \|x^{k+1} - x^k\|^2 / t_k \to 0$
- If $\sum_k t_k \to \infty$ the sequence is minimizing
- If $\{t_k\}$ bounded, the whole sequence converges to a minimizer and $f(x^k) \searrow \min f$

Proximal point algorithm: convergence

$$x^{k+1} = \arg\min f(y) + \frac{1}{2t_k}\|y - x^k\|_2^2$$

`(DR)` holds at all iterations

(sharper inequality without $\frac{1}{2}$ also holds)

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2t_k}\|x^{k+1} - x^k\|_2^2$$

- If $\arg\min f \neq \emptyset$ then $\delta_k = \|x^{k+1} - x^k\|^2 / t_k \to 0$
- If $\sum_k t_k \to \infty$ the sequence is minimizing
- If $\{t_k\}$ bounded, the whole sequence converges to a minimizer and $f(x^k) \searrow \min f$

Mathematical Programming

February 1993, Volume 62, Issue 1-3, pp 261-275

Convergence of some algorithms for convex minimization

Rafael Correa, Claude Lemaréchal

the proximal point subgradient

$$p = p_t^f(x) \iff p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2$$
$$\iff 0 \in \partial f(p) + \frac{1}{t}(p - x)$$

Take $g(p) := \frac{1}{t}(x - p) \in \partial f(p)$ satisfying the OC

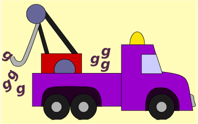
the proximal point subgradient

$$
\begin{aligned}
p = p_t^f(x) \quad &\iff \quad p = \arg\min f(y) + \frac{1}{2t}\|y - x\|_2^2 \\
&\iff \quad 0 \in \partial f(p) + \frac{1}{t}(p - x)
\end{aligned}
$$

Take $g(p) := \frac{1}{t}(x - p) \in \partial f(p)$ satisfying the OC

Proceeding like in the transportation formula,

$$
g(p) \in \partial_\varepsilon f(x) \quad \text{for } \varepsilon \;
\begin{aligned}
&:= \; f(x) - f(p) - g(p)^\top (x - p) \\
&= \; f(x) - f(p) - \|x - p\|^2 / t
\end{aligned}
$$



the proximal point subgradient

$$\begin{aligned} p = p_t^f(x) \quad &\Longleftrightarrow \quad p = \arg\min f(y) + \tfrac{1}{2t}\|y-x\|_2^2 \\ &\Longleftrightarrow \quad 0 \in \partial f(p) + \tfrac{1}{t}(p-x) \end{aligned}$$

Take $g(p) := \frac{1}{t}(x-p) \in \partial f(p)$ satisfying the OC

Proceeding like in the transportation formula,

$$\begin{aligned} g(p) \in \partial_\varepsilon f(x) \quad \text{for } \varepsilon \quad &:= \quad f(x) - f(p) - g(p)^\top(x-p) \\ &= \quad f(x) - f(p) - \|x-p\|^2/t \end{aligned}$$

For the PPA, this means that

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and $\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$

PPA: an implicit ε-subgradient descent method

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and

$$\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$$

PPA: an implicit ε-subgradient descent method

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and

$$\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$$

note 4: the method is still implicit, check ε_{k+1}

PPA: an implicit ε-subgradient descent method

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and

$$\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$$

note 4: the method is still implicit, check ε_{k+1}

What if p_t^f is not computable?

PPA: an implicit ε-subgradient descent method

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and

$$\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$$

note 4: the method is still implicit, check ε_{k+1}

What if p_t^f is not computable?

Can we make the method implementable?

YES!

PPA: an implicit ε-subgradient descent method

$$x^{k+1} = x^k - t_k g^k \quad \text{for } g^k \in \partial_{\varepsilon_{k+1}} f(x^k)$$

and

$$\varepsilon_{k+1} := f(x^k) - f(x^{k+1}) - \frac{\|x^k - x^{k+1}\|^2}{t_k}$$

note 4: the method is still implicit, check ε_{k+1}

What if p_t^f is not computable?

Can we make the method implementable?

YES!

Use the bundle ideas

to compute (ap)proximal points

Bundle methods prove most useful

In situations

- when the objective function is not available explicitly

and/or

- when we do not have access to the full subdifferential

and/or

- when calculations need to be done with high precision

Bundling to approximate the p

WANT: $$p = p_t^f(\hat{x}) = \arg\min f(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2$$

or, equivalently, finding p such that $\frac{1}{t}(\hat{x} - p) \in \partial f(p)$

Bundling to approximate the p

WANT: $$p = p_t^f(\hat{x}) = \arg\min f(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2$$

or, equivalently, finding p such that $\frac{1}{t}(\hat{x} - p) \in \partial f(p)$

HAVE: $\mathbf{M}$, a model of f for which we do have full knowledge of the subdifferential (recall note for future use):

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2 \qquad (\mathtt{QP})$$

or, equivalently, finding q such that $\frac{1}{t}(\hat{x} - q) \in \partial \mathbf{M}(q)$

Bundling to approximate the p

WANT: $$p = p_t^f(\hat{x}) = \arg\min f(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2$$

or, equivalently, finding p such that $\frac{1}{t}(\hat{x} - p) \in \partial f(p)$

HAVE: $\mathbf{M}$, a model of f for which we do have full knowledge of the subdifferential (recall note for future use):

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2 \qquad (\mathtt{QP})$$

or, equivalently, finding q such that $\frac{1}{t}(\hat{x} - q) \in \partial \mathbf{M}(q)$

Now the **explicit** inclusion can be solved!

Bundling to approximate the p

WANT: $$p = p_t^f(\hat{x}) = \arg\min f(y) + \frac{1}{2t}\|y-\hat{x}\|_2^2$$

or, equivalently, finding p such that $\frac{1}{t}(\hat{x}-p) \in \partial f(p)$

HAVE: **M**, a model of f for which we do have full knowledge of the subdifferential (recall note for future use):

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y-\hat{x}\|_2^2 \qquad (\mathtt{QP})$$

or, equivalently, finding q such that $\frac{1}{t}(\hat{x}-q) \in \partial \mathbf{M}(q)$

Now the **explicit** inclusion can be solved!

$$G := \frac{1}{t}(\hat{x}-q) = \sum_i \alpha^i g^i$$

is a convex combination of active subgradients, computed for free when solving (QP)

Bundling to approximate the p

Theorem[CL93] Suppose the models satisfy

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(q^k) + g(q^k)^\top (y - q^k)$
- $M_{k+1}(y) \geq M_k(q^k) + G^{k\top}(y - q^k)$

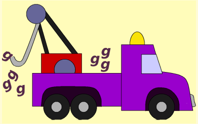
If $0 < t_{\min} \leq t_{k+1} \leq t_k$, then

$$\lim_{k\to\infty} q^k = \hat{x} \quad \text{and} \quad \lim_{k\to\infty} M_k(q^k) = f(\hat{x})$$

the bundle subgradient

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2 \qquad (\mathtt{QP})$$

Take $G = \frac{1}{t}(\hat{x} - q) \in \partial\mathbf{M}(q)$ satisfying the OC



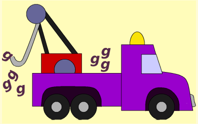
the bundle subgradient

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2 \qquad (\mathtt{QP})$$

Take $G = \frac{1}{t}(\hat{x} - q) \in \partial \mathbf{M}(q)$ satisfying the OC

Proceeding like in the transportation formula,

$$\begin{array}{llllll} G \in \partial_\varepsilon f(\hat{x}) & \text{for } \varepsilon & := & f(\hat{x}) - \mathbf{M}(q) & - & G^\top(\hat{x} - q) \\ & & = & f(\hat{x}) - \mathbf{M}(q) & - & \|\hat{x} - q\|^2/t \\ & & = & \delta & - & t\|G\|^2 \end{array}$$



the bundle subgradient

$$q = p_t^{\mathbf{M}}(\hat{x}) = \arg\min \mathbf{M}(y) + \frac{1}{2t}\|y - \hat{x}\|_2^2 \qquad (\mathtt{QP})$$

Take $G = \frac{1}{t}(\hat{x} - q) \in \partial \mathbf{M}(q)$ satisfying the OC

Proceeding like in the transportation formula,

$$\begin{array}{lcccc} G \in \partial_\varepsilon f(\hat{x}) \quad \text{for } \varepsilon & := & f(\hat{x}) - \mathbf{M}(q) & - & G^\top(\hat{x} - q) \\ & = & f(\hat{x}) - \mathbf{M}(q) & - & \|\hat{x} - q\|^2/t \\ & = & \delta & - & t\|G\|^2 \end{array}$$

For the bundling procedure, this means (now $q = x^{k+1}$)

$$x^{k+1} = \hat{x}^k - t_k G^k \quad \text{for } G^k \in \partial_{\varepsilon_k} f(\hat{x}^k)$$

and $\delta_k = \varepsilon_k + t_k\|G^k\|^2$ all **explicit** values, computed when solving (QP)

Illustration of oracles and models

The cutting-plane model is not the only one satisfying the [CL93] conditions

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(x^{k+1}) + g^{k+1\top}(y - x^{k+1})$
- $M_{k+1}(y) \geq M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})$

Illustration of oracles and models

The cutting-plane model is not the only one satisfying the [CL93] conditions

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(x^{k+1}) + g^{k+1\top}(y - x^{k+1})$
- $M_{k+1}(y) \geq M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})$

We can use

- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \leq k\right\}$
- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \in I(x^{k+1})\right\}$
- $M_{k+1}(y) =$ $\max\left\{f^{k+1} + g^{k+1\top}(y - x^{k+1}), M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})\right\}$
- Anything in-between

Illustration of oracles and models

The cutting-plane model is not the only one satisfying the [CL93] conditions

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(x^{k+1}) + g^{k+1\top}(y - x^{k+1})$
- $M_{k+1}(y) \geq M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})$

We can use

- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \leq k\right\}$
- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \in I(x^{k+1})\right\}$
- $M_{k+1}(y) =$
  $\max\left\{f^{k+1} + g^{k+1\top}(y - x^{k+1}), M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})\right\}$
- Anything in-between
  $M_{k+1}(y) = \max\left\{f^i + g(q^i)^\top(y - x^i) : i \in \mathcal{B}_k\right\}$

Illustration of oracles and models

The cutting-plane model is not the only one satisfying the [CL93] conditions

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(x^{k+1}) + g^{k+1\top}(y - x^{k+1})$
- $M_{k+1}(y) \geq M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})$

We can use

- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \leq k\right\}$
- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \in I(x^{k+1})\right\}$
- $M_{k+1}(y) = \max\left\{f^{k+1} + g^{k+1\top}(y - x^{k+1}), M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})\right\}$
- Anything in-between $M_{k+1}(y) = \max\left\{f^i + g(q^i)^\top(y - x^i) : i \in \mathcal{B}_k\right\}$

This allows to keep controlled the `(QP)` size

Illustration of oracles and models

The cutting-plane model is not the only one satisfying the [CL93] conditions

- $M_k(y) \leq f(y)$ for all k and y
- $M_{k+1}(y) \geq f(x^{k+1}) + g^{k+1\top}(y - x^{k+1})$
- $M_{k+1}(y) \geq M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})$

We can use

- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \leq k\right\}$
- $M_{k+1}(y) = \max\left\{f^i + g^{i\top}(y - x^i) : i \in I(x^{k+1})\right\}$
- $M_{k+1}(y) =$ $\max\left\{f^{k+1} + g^{k+1\top}(y - x^{k+1}), M_k(x^{k+1}) + G^{k\top}(y - x^{k+1})\right\}$
- Anything in-between $M_{k+1}(y) = \max\left\{f^i + g(q^i)^\top(y - x^i) : i \in \mathcal{B}_k\right\}$

This allows to keep controlled the `(QP)` size

There are many different models

CP models are the most straightforward ones. There are other possibilities, that exploit structural knowledge

There are many different models

CP models are the most straightforward ones. There are other possibilities, that exploit structural knowledge

For $x \in \mathbb{R}^n$, given matrices $A \succeq 0$, $B \succ 0$,

$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

is called the “half-and-half” function

There are many different models

CP models are the most straightforward ones. There are other possibilities, that exploit structural knowledge

For $x \in \mathbb{R}^n$, given matrices $A \succeq 0$, $B \succ 0$,

$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

is called the “half-and-half” function

For $n = 2$, it corresponds to

$$f(x_1, x_2) = a|x_1| + bx_2^2$$

There are many different models

CP models are the most straightforward ones. There are other possibilities, that exploit structural knowledge

For $x \in \mathbb{R}^n$, given matrices $A \succeq 0$, $B \succ 0$,

$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

is called the “half-and-half” function

For $n = 2$, it corresponds to

$$f(x_1, x_2) = a|x_1| + b x_2^2$$

The oracle information for f can be available in different forms

Models for the half-and-half function

STRUCTURE	$f(x)$	
none	$\sqrt{x^\top A x} + x^\top B x$	
sum	$f_1(x) + f_2(x)$	$f_1(x) = \sqrt{x^\top A x}$ $f_2(x) = x^\top B x$ f_2 is smooth
compo sition	$(h \circ c)(x)$	$c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1}$ c $h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1}$ h

Models for the half-and-half function

STRUCTURE	$f(x)$	
none	$\sqrt{x^\top Ax} + x^\top Bx$	
sum	$f_1(x) + f_2(x)$	$f_1(x) = \sqrt{x^\top Ax}$ $f_2(x) = x^\top Bx$ **f_2 is smooth**
compo sition	$(h \circ c)(x)$	$c(x) = (x, x^\top Bx) \in \mathbb{R}^{n+1}$ **c is smooth** $h(C) = \sqrt{C_{1:n}^\top AC_{1:n}} + C_{n+1}$ **h is sublinear**

Models for the half-and-half function

STRUCTURE	$f(x)$	
none	$\sqrt{x^\top Ax} + x^\top Bx$	$f^k := f(x^k),\ g^k \in \partial f(x^k)$
sum	$f_1(x) + f_2(x)$	$f_1(x) = \sqrt{x^\top Ax}$ $f_2(x) = x^\top Bx$
compo sition	$(h \circ c)(x)$	$c(x) = (x, x^\top Bx) \in \mathbb{R}^{n+1}$ $h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1}$

Models for the half-and-half function

STRUCTURE	$f(x)$	
none	$\sqrt{x^\top A x} + x^\top B x$	$f^k := f(x^k),\ g^k \in \partial f(x^k)$
sum	$f_1(x) + f_2(x)$	$f_1(x) = \sqrt{x^\top A x}$ $f_2(x) = x^\top B x$ $f_1^k, g_1^k, f_2^k, \nabla f_2(x^k)$
compo sition	$(h \circ c)(x)$	$c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1}$ $h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1}$

Models for the half-and-half function

STRUCTURE	$f(x)$	
none	$\sqrt{x^\top A x} + x^\top B x$	$f^k := f(x^k),\ g^k \in \partial f(x^k)$
sum	$f_1(x) + f_2(x)$	$f_1(x) = \sqrt{x^\top A x}$ $f_2(x) = x^\top B x$ $f_1^k, g_1^k, f_2^k, \nabla f_2(x^k)$
compo sition	$(h \circ c)(x)$	$c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1}$ $c^k = c(x^k), c'(x^k)$ $h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1}$ $h^k, g^k \in \partial h(c^k)$

Take away messages

- PPA has good properties for minimizing convex nonsmooth functions
- If computing exactly the prox at each iteration is too demanding, a bundling approach can be put in place
- the theory in [CL93]] allows for great generality in how the approximation is done
- Bundle methods enter into play to decide when the approximal point is sufficiently good `(DR)`.
- If rough approximations without optimality certificate are enough: SG
- CP methods are more reliable (stopping test) but LP grows indefinitely, need bounding set X, and can be unstable
- Bundle methods stabilize CP, have reliable stopping test, solve one QP of controllable size per iteration

Take away messages

- PPA has good properties for minimizing convex nonsmooth functions
- If computing exactly the prox at each iteration is too demanding, a bundling approach can be put in place
- the theory in [CL93]] allows for great generality in how the approximation is done
- Bundle methods enter into play to decide when the approximal point is sufficiently good `(DR)`.
- If rough approximations without optimality certificate are enough: SG
- CP methods are more reliable (stopping test) but LP grows indefinitely, need bounding set X, and can be unstable
- Bundle methods stabilize CP, have reliable stopping test, solve one QP of controllable size per iteration

Beware all of the above depends on having exact oracle information