ALGORITHMS FOR TWO-STAGE SP: A PRIMER ON NONSMOOTH OPTIMIZATION

Claudia Sagastizábal

BAS Lecture 9, April 7, 2016, IMPA

SVAN 2016
Set YouTube resolution to 480p for best viewing
Two-Stage LP with RCR\textsuperscript{a}, $\Omega = \{\omega^1, \ldots, \omega^S\}$

$$\min_{x \in X} c^\top x + \phi(x) \quad \text{for} \quad X := \{x \geq 0 : Ax = b\},$$

where $\phi(x) = \mathbb{E} \left[ Q(x, \xi) \right] = \sum_{s=1}^{S} p_s Q(x, \xi^s)$ and

$$Q(x, \xi^s) = \begin{cases} \min & q^s \top y \\ \text{s.t.} & W^s y = h^s - T^s x \\ & y \geq 0 \end{cases} = \begin{cases} \max & \pi \top (h^s - T^s x) \\ \text{s.t.} & W^s \top \pi \leq q^s \end{cases}$$

$$\partial \phi(x^k) = - \sum_{s=1}^{S} p_s T^s \top \arg \max \left\{ \pi \top (h^s - T^s x) : \pi \in \Pi(q^s) \right\}$$

\textsuperscript{a}next lecture: without Relative Complete Recourse (infeasibility!)
Two-Stage LP with RCR\textsuperscript{a}, \( \Omega = \{\omega^1, \ldots, \omega^S\} \)

\[
\min_{x \in X} \mathbf{c}^\top x + \phi(x) \quad \text{for} \quad X := \{x \geq 0 : \mathbf{A}x = \mathbf{b}\},
\]

where
\[
\phi(x^k) = \mathbb{E} \left[ Q(x^k, \xi) \right] = \sum_{s=1}^{S} p_s Q(x^k, \xi^s)
\]
and
\[
Q(x^k, \xi^s) = \begin{cases} 
\min & q^s \top y \\
\text{s.t.} & W^s y = h^s - T^s x^k \\
& y \geq 0
\end{cases}
\]

\[
\partial \phi(x^k) = -\sum_{s=1}^{S} p_s T^s \top \arg \max \left\{ \pi^\top (h^s - T^s x^k) : \pi \in \Pi(q^s) \right\}
\]

\textsuperscript{a}next lecture: without Relative Complete Recourse (infeasibility!)
Two-Stage LP with RCR, $\Omega = \{\omega^1, \ldots, \omega^S\}$

$$\min_{x \in X} c^\top x + \phi(x) \quad \text{for} \quad X := \{x \geq 0 : Ax = b\},$$

where $\phi(x^k) = \mathbb{E} \left[ Q(x^k, \xi) \right] = \sum_{s=1}^{S} p_s Q(x^k, \xi^s)$ and

$$Q(x^k, \xi^s) = \begin{cases} \min & q^s \top y \\ \text{s.t.} & W^s y = h^s - T^s x^k \\ & y \geq 0 \end{cases} = \begin{cases} \max & \pi^\top (h^s - T^s x^k) \\ \text{s.t.} & W^s \top \pi \leq q^s \end{cases}$$

Primal solution $y^{s,k}$

Dual solution $\pi^{s,k}$

$$\partial \phi(x^k) = -\sum_{s=1}^{S} p_s T^s \top \arg \max \left\{ \pi^\top (h^s - T^s x^k) : \pi \in \Pi(q^s) \right\}$$
Evaluating $\phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k} \top (h^s - T^s x^k)$
Evaluating \( \phi(x^k) = \sum_{s=1}^{S} p_s \pi_s^{k} \top (h^s - T^s x^k) \) gives for free a subgradient \( \gamma^k = - \sum_{s=1}^{S} p_s T^s \top \pi_s^{s,k} \in \partial \phi(x^k) \)
Evaluating $\phi(x^k) = \sum_{s=1}^S p_s \pi^{s,k} \top (h^s - T^s x^k)$ gives for free a subgradient $
abla^k = -\sum_{s=1}^S p_s T^s \top \pi^{s,k} \in \partial \phi(x^k)$ and the linearization

$$\geq \phi(x^k) + \nabla^k \top (x - x^k)$$

$$= \sum_{s=1}^S p_s \pi^{s,k} \top (h^s - T^s x^k) - \sum_{s=1}^S p_s \pi^{s,k} \top T^s (x - x^k)$$

$$= \sum_{s=1}^S p_s \pi^{s,k} \top (h^s - T^s x)$$
Evaluating \( \phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k} \top (h^s - T^s x^k) \)

gives for free a subgradient \( \gamma^k = -\sum_{s=1}^{S} p_s T^s \top \pi^{s,k} \in \partial \phi(x^k) \) and

the linearization

\[
\phi(x) \geq \phi(x^k) + \gamma^k \top (x - x^k) = \sum_{s=1}^{S} p_s \pi^{s,k} \top (h^s - T^s x^k) - \sum_{s=1}^{S} p_s \pi^{s,k} \top T^s (x - x^k) = \sum_{s=1}^{S} p_s \pi^{s,k} \top (h^s - T^s x)
\]
Evaluating $\phi$ at $x^k$

1st-stage problem

\[ \min c^T x + \phi(x) \quad x \in X \]

2nd-stage subproblems

- $Q(x^k, \xi^1)$
- $\phi(x^k)$ and a subgradient
- $Q(x^k, \xi^S)$
Evaluating $\phi$ at $x^k$
gives the linearization:

$$\phi(x) \geq \sum_{s=1}^{S} p_s \pi_{s,k}^\top (h^s - T^s x)$$

1st-stage problem:

$$\min c^\top x + \phi(x)$$
$$x \in X$$

2nd-stage subproblems:

$$Q(x^k, \xi^1)$$

and a subgradient:

$$\phi(x^k)$$

$$Q(x^k, \xi^S)$$
Evaluating $\phi$ at $x^k$

gives the linearization $\phi(x) \geq \sum_{s=1}^{S} (p_s \pi_s, k^\top (h_s - T_s x))$

1st-stage problem

$$\min_{x \in X} c^T x + \phi(x)$$

minimize convex nonsmooth knowing $f(x)$ and $g(x) \in \partial f(x)$ (one)
Computational NSO: what does it mean?

For the unconstrained\(^a\) problem

\[
\min f(x),
\]

where \(f\) is convex but not differentiable at some points

\(^a\mathbb{X} = \mathbb{R}^n\) today
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points, we shall define **algorithms** based on information provided by an oracle or “black box”

\[ x \rightarrow f(x) \rightarrow g(x) \in \partial f(x) \]
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

What do we mean by an algorithm?

An example

[source http://comofas.com/]
What do we mean by an algorithm?

An example
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?
What do we mean by an algorithm?

repeat until ... ??
What do we mean by an algorithm?

An algorithm is a sequence of steps that are repeated until satisfaction.
What do we mean by an algorithm?

An algorithm is a sequence of steps that are repeated until satisfaction of a stopping test.
Back to Computational NSO

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points, we look for algorithms based on information provided by an oracle or “black box”

endowed with reliable stopping tests
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function

\[ \partial f(x) = \{ \nabla f(x) \} = \{ \text{slopes of linearizations supporting } f, \text{ tangent at } x \} \]
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function
What can be done with the oracle information?

An example of a convex nonsmooth function

\[ \partial f(x) = \{ g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y \} \]
What can be done with the oracle information?

An example of a convex nonsmooth function

\[ \partial f(x) = \{ g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top(y - x) \text{ for all } y \} \]
\[ = \{ \text{slopes of linearizations supporting } f, \text{ tangent at } x \} \]
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad \leftrightarrow \quad |g(x^k)| \leq \text{TOL} \]
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ \nabla f(x^k) = 1, \quad \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods **do not work**

\[
|\nabla f(x^k)| = 1, \ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL)\)
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL) \]
Why special NSO methods?

Smooth optimization methods **do not work**

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

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\)
Why special NSO methods?

Smooth optimization methods **do not work**

\[
f(x) = |x|
\]

\[
|\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\)
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods **do not work**

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

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\)
Why special NSO methods?

Smooth optimization methods **do not work**

\[
f(x) = |x|
\]

\[
|\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL)\)
Why special NSO methods?

Smooth optimization methods **do not work**

\[
f(x) = |x|
\]

\[
|\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\]

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

Finite difference approximations **fail** (no automatic differentiation)
Why special NSO methods?

Smooth optimization methods **do not work**

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

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

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

\(-g(x^k)\) may **not** provide descent
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x_k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \(|\nabla f(x_k)| \leq TOL \quad (\leftrightarrow |g(x_k)| \leq TOL)\)

Finite difference approximations fail

Linesearches get trapped in kinks and fail

\(-g(x^k)\) may not provide descent
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.
How is the oracle information used?

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

Subgradient 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 $x^{k+1} = x^k - t_k \frac{g(x^k)}{\|g(x^k)\|}$ 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 $x^{k+1} = x^k - t_k \frac{g(x^k)}{\|g(x^k)\|}$ for a suitable stepsize $t_k > 0$.
3. Make $k = k + 1$ and loop to 1.

Is this a good “recipe”?
Subgradient Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} = x^k - t_k \frac{g(x^k)}{\|g(x^k)\|}$ for a suitable stepsize $t_k > 0$.
3 Make $k = k + 1$ and loop to 1.

SG methods are the algorithmic version of this road sign.
Subgradient Methods

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

1 Call the oracle at $x^k$.

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

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

SG methods are
the algorithmic version
of this road sign

...something is missing!!!
Subgradient Methods

0 Choose \( x^1 \) and set \( k = 1 \).
1 Call the oracle at \( x^k \).
2 Compute \( x^{k+1} = x^k - t_k \frac{g(x^k)}{\|g(x^k)\|} \) for a suitable stepsize \( t_k > 0 \).
3 Make \( k = k + 1 \) and loop to 1.

SG methods are
the algorithmic version
of this road sign
not a good recipe
Subgradient Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} = x^k - t_k \frac{g(x^k)}{\|g(x^k)\|}$ for a suitable stepsize $t_k > 0$.
3 Make $k = k + 1$ and loop to 1.

SG methods are not a good recipe
the algorithmic version of this road sign

Non-monotone!
Subgradient Methods: why a “not-good” recipe

Non-monotone functional values, but converges

because distance to solution set decreases for $\sum t_k = +\infty, \sum t_k^2 < +\infty$
Subgradient Methods: why a “not-good” recipe

\[
\begin{cases}
\text{Non-monotone functional values, but converges} \\
\text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty
\end{cases}
\]

Constrained case dealt with by projecting onto \( X \): reasonable for simple \( X \) only
Subgradient Methods: why a “not-good” recipe

\[
\begin{align*}
\text{Non-monotone functional values, but converges} \\
\text{because distance to solution set decreases for } & \sum t_k = +\infty, \sum t_k^2 < +\infty \\
\text{Lacks a stopping test}
\end{align*}
\]
Subgradient Methods: why a “not-good” recipe

- Non-monotone functional values, but converges
- Because distance to solution set decreases for $\sum t_k = +\infty, \sum t_k^2 < +\infty$
- Lacks a stopping test

... does not use all available information
Subgradient Methods: why a “not-good” recipe

\[ \begin{align*}
&\text{Non-monotone functional values, but converges} \\
&\text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty \\
&\text{Lacks a stopping test}
\end{align*} \]

…does not use all \textcolor{red}{\text{available}} information

\[ x \quad \xrightarrow{\quad} \quad f(x) \quad \xleftarrow{\quad} \quad g(x) \in \partial f(x) \]
Subgradient Methods: why a “not-good” recipe

Non-monotone functional values, but converges
because distance to solution set decreases for $\sum t_k = +\infty, \sum t_k^2 < +\infty$

Lacks a stopping test

...does not use all available information

SG methods are like caipirinha without cachaça
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 reliable stopping test.
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 $f$.

$$
\begin{align*}
    x_i & \quad f^i = f(x^i) \\
    g^i & = g(x^i) \\
    f^i + g^i\top (x - x^i)
\end{align*}
$$
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 $f$.

$$x^i \quad \begin{align*}
    f^i &= f(x^i) \\
    g^i &= g(x^i)
\end{align*}$$

$$\implies M(x) = \max_i \{ f^i + g^i \top (x - x^i) \}$$
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 $f$.

$$
\begin{align*}
    x^i & \quad f^i = f(x^i) \\
    g^i & = g(x^i) \\
\end{align*}
$$

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

(just an example, many other models are possible)
Cutting-plane methods

To minimize $f$ (unavailable in an explicit manner), minimize its model $M(x) = \max_{i} \left\{ f^i + g^i (x - x^i) \right\}$

Improve the model at each iteration
Cutting-plane methods

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

Improve the model at each iteration:

$$M_{k+1}(x) = \max_{i \leq k+1} \left\{ f^i + g^i \top (x - x^i) \right\} = \max \left( M_k(x), f^{k+1} + g^{k+1 \top} (x - x^{k+1}) \right)$$

where $x^{k+1}$ minimizes $M_k$
Cutting-plane methods

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

Improve the model at each iteration:

\[
M_{k+1}(x) = \max_{i \leq k+1} \left\{ f^i + g^{i \top} (x - x^i) \right\} \\
= \max \left( M_k(x), f^{k+1} + g^{k+1 \top} (x - x^{k+1}) \right)
\]

where $x^{k+1}$ minimizes $M_k$

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:

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

where $x^{k+1}$ minimizes $\mathbf{M}_k$

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

$x^{k+1} \in \arg \min \mathbf{M}_k(x)$ iteratively
Cutting-plane methods

Artificial bounding at least for the first iterations
Cutting-plane methods

\[ f(x) \]

\[ x^1 \]

\[ X \]
Cutting-plane methods

\[ f(x) \]

\[ X \]

\[ x^1 \]

\[ x^2 \]
Cutting-plane methods
Cutting-plane methods

\[ f(x) \]

\[ x^1 \quad x^4 \quad x^3 \quad x^2 \]

\[ X \]
Cutting-plane methods

\[ f(x) \]

\[
\begin{align*}
X &= \{ x^1, x^5, x^4, x^3, x^2 \} \\
\end{align*}
\]
Cutting-plane methods

\{M_k(x^{k+1})\} increases
Cutting-plane methods

\[ \{ M_k(x^{k+1}) \} \text{ increases but not necessarily the functional values:} \]
\[ f(x^5) > f(x^4) \]
{M_k(x^{k+1})} increases but not necessarily the functional values: \( f(x^5) > f(x^4) \). **Stopping test measures** \( \delta_k := f(x^k) - M_{k-1}(x^k) \)
Cutting-plane Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} \in \arg\min_x M_k(x)$
3 $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^{k^\top}(\cdot - x^k) \right)$, $k = k + 1$, loop to 1.
Cutting-plane Methods

0 Choose \( x^1 \) and set \( k = 1 \).
1 Call the oracle at \( x^k \). \textbf{If} \( f(x^k) - M_{k-1}(x^k) \leq \text{tol} \) \textit{STOP}
2 Compute \( x^{k+1} \in \arg\min_x M_k(x) \)
3 \( M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^k(\cdot - x^k) \right) \), \( k = k + 1 \), loop to 1.

CP methods are
an improved algorithmic version
of the Aussie sign

a better recipe