A few constructive approaches to optimal first-order optimization methods for convex optimization

Adrien Taylor

All Russian optimization seminar – May 2021

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Based on original ideas by Drori and Teboulle (2014).

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Based on original ideas by Drori and Teboulle (2014).

My personal (and informal) view on this topic

based on insights obtained through works with great collaborators.

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Based on original ideas by Drori and Teboulle (2014).

My personal (and informal) view on this topic

based on insights obtained through works with great collaborators.

Informal and example-based presentation.

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Based on original ideas by Drori and Teboulle (2014).

My personal (and informal) view on this topic

based on insights obtained through works with great collaborators.

Informal and example-based presentation.

If interested, details are provided in references at the end.

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

- Based on original ideas by Drori and Teboulle (2014).
- My personal (and informal) view on this topic
  - based on insights obtained through works with great collaborators.

Informal and example-based presentation.

- If interested, details are provided in references at the end.
- Complementary material on Francis Bach's blog (also $\pm$ informal)
  - `https://francisbach.com/computer-aided-analyses/`

Disclaimers about this presentation

Overall idea: principled approach to worst-case analyses in first-order optimization.

Based on original ideas by Drori and Teboulle (2014).

My personal (and informal) view on this topic

based on insights obtained through works with great collaborators.

Informal and example-based presentation.

If interested, details are provided in references at the end.

Complementary material on Francis Bach's blog (also $\pm$ informal)

`https://francisbach.com/computer-aided-analyses/`

More examples in toolbox' manual

`https://github.com/AdrienTaylor/Performance-Estimation-Toolbox`

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

'14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.

Genealogy on this topic (“my humble, biased, view on...”)

Base methodological developments:

'14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.

'16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

'14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.

'16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.

'16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints ("IQCs"). Essentially Lyapunov functions.

In this presentation:

Genealogy on this topic (“my humble, biased, view on...”)

Base methodological developments:

- ’14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.
- ’16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.
- ’16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints (“IQCs”). Essentially Lyapunov functions.

In this presentation:

- ’17 T, Hendrickx and Glineur: interpolation (tightness), and primal/dual interpretations of the SDPs, and few generalizations of the approach.

Genealogy on this topic (“my humble, biased, view on...”)

Base methodological developments:

- '14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.
- '16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.
- '16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints (“IQCs”). Essentially Lyapunov functions.

In this presentation:

- '17 T, Hendrickx and Glineur: interpolation (tightness), and primal/dual interpretations of the SDPs, and few generalizations of the approach.
- '20-'21 Drori & T.: design of optimal methods via minimax

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

- '14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.
- '16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.
- '16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints ("IQCs"). Essentially Lyapunov functions.

In this presentation:

- '17 T, Hendrickx and Glineur: interpolation (tightness), and primal/dual interpretations of the SDPs, and few generalizations of the approach.
- '20-'21 Drori & T.: design of optimal methods via minimax

But also:

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

- '14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.
- '16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.
- '16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints ("IQCs"). Essentially Lyapunov functions.

In this presentation:

- '17 T, Hendrickx and Glineur: interpolation (tightness), and primal/dual interpretations of the SDPs, and few generalizations of the approach.
- '20-'21 Drori & T.: design of optimal methods via minimax

But also:

- ⋄ Fair amount of algorithmic analyses (and design) originated from SDPs (from different authors, examples below), in different settings.

Genealogy on this topic ("my humble, biased, view on...")

Base methodological developments:

'14 Drori and Teboulle (MP): upper bounds on worst-case behaviors of FO methods via SDPs, idea of using this machinery for designing methods.

'16 Kim and Fessler (MP): design of an optimized method for smooth convex minimization, using SDPs.

'16 Lessard, Recht, Packard (SIOPT): smaller SDPs for linear convergence, via integral quadratic constraints ("IQCs"). Essentially Lyapunov functions.

In this presentation:

'17 T, Hendrickx and Glineur: interpolation (tightness), and primal/dual interpretations of the SDPs, and few generalizations of the approach.

'20-'21 Drori & T.: design of optimal methods via minimax

But also:

- ◇ Fair amount of algorithmic analyses (and design) originated from SDPs (from different authors, examples below), in different settings.
- ◇ We try keeping track of related works in the toolbox' manual (see later), incomplete references in this presentation.

François Glineur

Julien Hendrickx

Etienne de Klerk

Ernest Ryu

Yoel Drori

Francis Bach

Jérôme Bolte

Alexandre d'Aspremont

Mathieu Barré

Radu-Alexandru Dragomir

Bryan Van Scoy

Laurent Lessard

Carolina Bergeling

Pontus Giselsson

Take-home messages

Worst-cases are solutions to optimization problems.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

When it does: principled approach to worst-case analyses.

Performance estimation problems

Designing methods using PEPs

Conclusions

Performance estimation problems

Designing methods using PEPs

Conclusions

Analysis of a gradient method

Say we aim to solve

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

under some assumptions on f (it belongs to some class of functions).

Analysis of a gradient method

Say we aim to solve

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

under some assumptions on f (it belongs to some class of functions).

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

Analysis of a gradient method

Say we aim to solve

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

under some assumptions on f (it belongs to some class of functions).

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

Question: what *a priori* guarantees after N iterations?

Analysis of a gradient method

Say we aim to solve

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

under some assumptions on f (it belongs to some class of functions).

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

Question: what *a priori* guarantees after N iterations?

Examples: how small should $f(x_N) - f(x_\star)$, $\|f'(x_N)\|$, $\|x_N - x_\star\|$ be?

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- ◇ for all f satisfying some assumptions,

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- for all f satisfying some assumptions,
- for x_N was obtained through gradient descent from x_0?

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- $\diamond$ for all f satisfying some assumptions,
- $\diamond$ for x_N was obtained through gradient descent from x_0?

By definition, the "best" such guarantee is

$$\left\|f'(x_N)\right\| \leqslant \text{"worst possible value of } \left\|f'(x_N)\right\|\text{, given the assumptions".}$$

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- for all f satisfying some assumptions,
- for x_N was obtained through gradient descent from x_0?

By definition, the "best" such guarantee is

$$\left\|f'(x_N)\right\| \leqslant \text{“worst possible value of } \left\|f'(x_N)\right\|\text{, given the assumptions”.}$$

In other words:

$$\begin{aligned} \left\|f'(x_N)\right\| \leqslant \sup_{F, y_0, \ldots, y_N} \quad & \left\|F'(y_N)\right\| \\ \text{subject to} \quad & y_1, \ldots, y_N \text{ generated by gradient method from } y_0 \\ & F \text{ satisfies the assumptions on } f \end{aligned}$$

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- $\diamond$ for all f satisfying some assumptions,
- $\diamond$ for x_N was obtained through gradient descent from x_0?

By definition, the "best" such guarantee is

$$\left\|f'(x_N)\right\| \leqslant \text{"worst possible value of } \left\|f'(x_N)\right\|\text{, given the assumptions".}$$

In other words:

$$\begin{aligned} \left\|f'(x_N)\right\| \leqslant \sup_{F, y_0, \ldots, y_N} \quad & \left\|F'(y_N)\right\| \\ \text{subject to} \quad & y_1, \ldots, y_N \text{ generated by gradient method from } y_0 \\ & F \text{ satisfies the assumptions on } f \end{aligned}$$

This problem is typically unbounded (arbitrarily bad starting point are feasible).

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- $\diamond$ for all f satisfying some assumptions,
- $\diamond$ for x_N was obtained through gradient descent from x_0?

By definition, the "best" such guarantee is

$$\left\|f'(x_N)\right\| \leqslant \text{"worst possible value of } \left\|f'(x_N)\right\|\text{, given the assumptions".}$$

In other words:

$$\begin{aligned} \left\|f'(x_N)\right\| \leqslant \sup_{F, y_0, \ldots, y_N} \quad & \left\|F'(y_N)\right\| \\ \text{subject to} \quad & y_1, \ldots, y_N \text{ generated by gradient method from } y_0 \\ & F \text{ satisfies the assumptions on } f \end{aligned}$$

This problem is typically unbounded (arbitrarily bad starting point are feasible).

Standard workaround: assume something on the starting point,
for example: assume bounded $\|x_0 - x_\star\|^2$, $\|f'(x_0)\|^2$ or $f(x_0) - f(x_\star)$.

Worst-case guarantees

Example: what can we a priori guarantee on $\|f'(x_N)\|$

- $\diamond$ for all f **and all x_0** satisfying some assumptions,
- $\diamond$ for x_N was obtained through gradient descent from x_0?

By definition, the "best" such guarantee is

$$\|f'(x_N)\| \leqslant \text{"worst possible value of } \|f'(x_N)\|\text{, given the assumptions".}$$

In other words:

$$\begin{aligned} \|f'(x_N)\| \leqslant \sup_{F, y_0, \ldots, y_N} \quad & \|F'(y_N)\| \\ \text{subject to} \quad & y_1, \ldots, y_N \text{ generated by gradient method from } y_0 \\ & F \text{ satisfies the assumptions on } f \\ & y_0 \textbf{ not too bad.} \end{aligned}$$

This problem is typically unbounded (arbitrarily bad starting point are feasible).

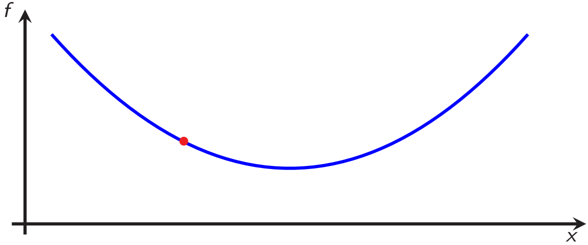
Standard workaround: assume something on the starting point,
for example: assume bounded $\|x_0 - x_\star\|^2$, $\|f'(x_0)\|^2$ or $f(x_0) - f(x_\star)$.

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:

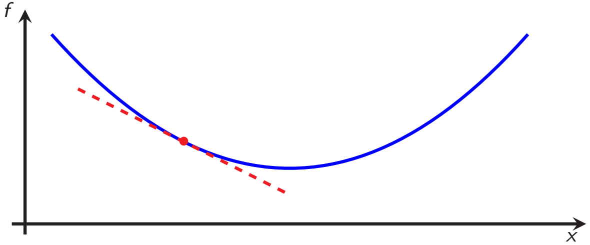
Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



Smooth strongly convex functions

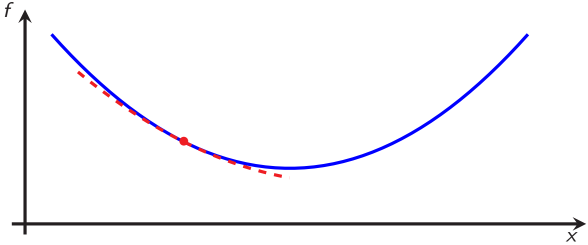
Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:

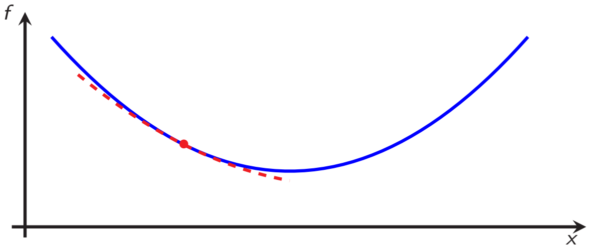


(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2$,

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



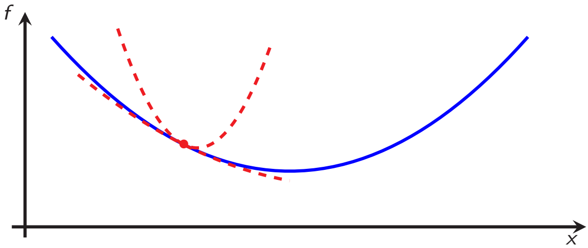
(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2}\|x - y\|^2$,

(2) (L-smoothness) $f(x) \leqslant f(y) + \langle f'(y), x - y \rangle + \frac{L}{2}\|x - y\|^2$.

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2$,

(2) (L-smoothness) $f(x) \leqslant f(y) + \langle f'(y), x - y \rangle + \frac{L}{2} \|x - y\|^2$.

Convergence rate of a gradient step

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathsf{L}}$),
- x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\|f'(x_0)\|$ is bounded?

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- $\diamond$ f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathbf{L}}$),
- $\diamond$ x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\diamond$ $\|f'(x_0)\|$ is bounded?

$$\max_{f, x_0, x_1} \left\|f'(x_1)\right\|^2$$

$$\text{s.t. } f \in \mathcal{F}_{\mu,L} \qquad\qquad \text{Functional class}$$

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathbf{L}}$),
- x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\|f'(x_0)\|$ is bounded?

$$
\begin{aligned}
\max_{f, x_{\mathbf{0}}, x_{\mathbf{1}}} \quad & \left\|f'(x_1)\right\|^2 & \\
\text{s.t.} \quad & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm}
\end{aligned}
$$

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- $\diamond$ f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathbf{L}}$),
- $\diamond$ x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\diamond$ $\|f'(x_0)\|$ is bounded?

$$
\begin{aligned}
\max_{f, x_0, x_1} \quad & \left\|f'(x_1)\right\|^2 & \\
\text{s.t.} \quad & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathbf{L}}$),
- x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\|f'(x_0)\|$ is bounded?

$$
\begin{aligned}
\max_{f, x_0, x_1} \quad & \left\|f'(x_1)\right\|^2 & \\
\text{s.t.} \quad & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1;

Convergence rate of a gradient step

Toy example: What can we guarantee on $\|f'(x_1)\|$ given that:

- $\diamond$ f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,\mathbf{L}}$),
- $\diamond$ x_1 was generated by gradient descent: $x_1 = x_0 - \gamma f'(x_0)$,
- $\diamond$ $\|f'(x_0)\|$ is bounded?

$$
\begin{aligned}
\max_{f, x_0, x_1} \quad & \left\| f'(x_1) \right\|^2 & \\
\text{s.t.} \quad & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\| f'(x_0) \right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1; parameters: μ, L, γ, R.

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

Idea:

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

Idea:

- replace f by its discrete version:

$$f_i = f(x_i),\ \ g_i = f'(x_i) \quad \forall i \in \{0, 1\}\,.$$

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

Idea:

- replace f by its discrete version:

$$f_i = f(x_i),\ \ g_i = f'(x_i) \quad \forall i \in \{0, 1\}\,.$$

- Require points (x_i, g_i, f_i) to be interpolable by a function $f \in \mathcal{F}_{\mu,L}$.

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

Idea:

- replace f by its discrete version:

$$f_i = f(x_i),\ g_i = f'(x_i) \quad \forall i \in \{0, 1\}\,.$$

- Require points (x_i, g_i, f_i) to be interpolable by a function $f \in \mathcal{F}_{\mu,L}$.
  The new constraint is:

$$\exists f \in \mathcal{F}_{\mu,L}:\ f_i = f(x_i),\ g_i = f'(x_i), \qquad \forall i \in \{0, 1\}\,.$$

Sampled version

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1} \quad & \|f'(x_1)\|^2 \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, \\ & x_1 = x_0 - \gamma f'(x_0), \\ & \|f'(x_0)\|^2 = R^2. \end{aligned}$$

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1} \quad & \|f'(x_1)\|^2 \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, \\ & x_1 = x_0 - \gamma f'(x_0), \\ & \|f'(x_0)\|^2 = R^2. \end{aligned}$$

- Variables: f, x_0, x_1.

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1} \quad & \|f'(x_1)\|^2 \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, \\ & x_1 = x_0 - \gamma f'(x_0), \\ & \|f'(x_0)\|^2 = R^2. \end{aligned}$$

- Variables: f, x_0, x_1.
- Sampled version:

$$\begin{aligned} \max_{\substack{x_0, x_1, g_0, g_1 \\ f_0, f_1}} \quad & \|g_1\|^2 \\ \text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0,1 \\ g_i = f'(x_i) & i = 0,1 \end{cases} \\ & x_1 = x_0 - \gamma g_0, \\ & \|g_0\|^2 = R^2. \end{aligned}$$

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1} \quad & \|f'(x_1)\|^2 \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, \\ & x_1 = x_0 - \gamma f'(x_0), \\ & \|f'(x_0)\|^2 = R^2. \end{aligned}$$

- Variables: f, x_0, x_1.
- Sampled version:

$$\begin{aligned} \max_{\substack{x_0, x_1, g_0, g_1 \\ f_0, f_1}} \quad & \|g_1\|^2 \\ \text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, 1 \\ g_i = f'(x_i) & i = 0, 1 \end{cases} \\ & x_1 = x_0 - \gamma g_0, \\ & \|g_0\|^2 = R^2. \end{aligned}$$

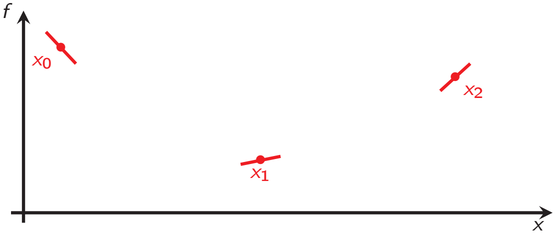
- Variables: x_0, x_1, g_0, g_1, f_0, f_1.

Smooth strongly convex interpolation

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.

Smooth strongly convex interpolation

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.

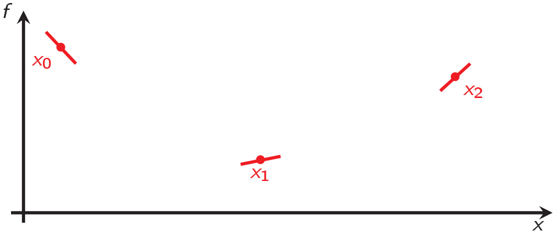


? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

Smooth strongly convex interpolation

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.



? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

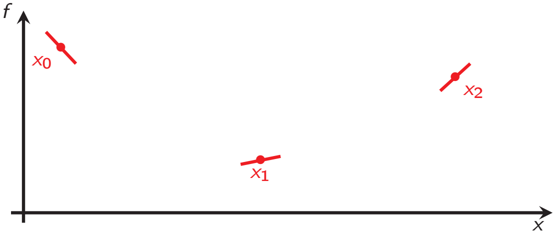
$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

- Necessary and sufficient condition: $\forall i,j \in S$

$$f_i \geqslant f_j + \langle g_j, x_i - x_j \rangle + \frac{1}{2L}\left\|g_i - g_j\right\|^2 + \frac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \frac{1}{L}(g_i - g_j)\right\|^2.$$

Smooth strongly convex interpolation

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.



? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

- Necessary and sufficient condition: $\forall i, j \in S$

$$f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\left\|g_i - g_j\right\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \tfrac{1}{L}(g_i - g_j)\right\|^2.$$

- Simpler example: pick $\mu = 0$ and $L = \infty$ (just convexity):

$$f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle.$$

Replace constraints

Replace constraints

◇ Interpolation conditions allow removing red constraints

$$
\begin{aligned}
\max_{\substack{x_0, x_1, g_0, g_1 \\ f_0, f_1}} \quad & \|g_1\|^2 \\
\text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, 1 \\ g_i = f'(x_i) & i = 0, 1 \end{cases} \\
& x_1 = x_0 - \gamma g_0, \\
& \|g_0\|^2 = R^2.
\end{aligned}
$$

Replace constraints

◇ Interpolation conditions allow removing red constraints

$$\begin{aligned}
\max_{\substack{x_0,x_1,g_0,g_1\\ f_0,f_1}} \quad & \|g_1\|^2 \\
\text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0,1 \\ g_i = f'(x_i) & i = 0,1 \end{cases} \\
& x_1 = x_0 - \gamma g_0, \\
& \|g_0\|^2 = R^2.
\end{aligned}$$

◇ replacing them by

$$f_1 \geqslant f_0 + \langle g_0, x_1 - x_0 \rangle + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_1 - x_0 - \tfrac{1}{L}(g_1 - g_0)\right\|^2$$

$$f_0 \geqslant f_1 + \langle g_1, x_0 - x_1 \rangle + \tfrac{1}{2L}\|g_0 - g_1\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(g_0 - g_1)\right\|^2.$$

Replace constraints

- Interpolation conditions allow removing red constraints

$$
\begin{array}{ll}
\displaystyle\max_{\substack{x_0, x_1, g_0, g_1\\ f_0, f_1}} & \|g_1\|^2 \\
\text{subject to} & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0,1 \\ g_i = f'(x_i) & i = 0,1 \end{cases} \\
& x_1 = x_0 - \gamma g_0, \\
& \|g_0\|^2 = R^2.
\end{array}
$$

- replacing them by

$$
f_1 \geqslant f_0 + \langle g_0, x_1 - x_0 \rangle + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_1 - x_0 - \tfrac{1}{L}(g_1 - g_0)\right\|^2
$$

$$
f_0 \geqslant f_1 + \langle g_1, x_0 - x_1 \rangle + \tfrac{1}{2L}\|g_0 - g_1\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(g_0 - g_1)\right\|^2.
$$

- Same optimal value (no relaxation); but still non-convex quadratic problem.

Semidefinite lifting

Semidefinite lifting

◇ Using $x_1 = x_0 - \gamma g_0$, all elements are quadratic in (g_0, g_1), and linear in (f_0, f_1):

$$
\begin{aligned}
\max_{\substack{g_0, g_1 \\ f_0, f_1}} \quad & \|g_1\|^2 \\
\text{subject to} \quad & f_1 \geqslant f_0 - \gamma\|g_0\|^2 + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\gamma g_0 + \tfrac{1}{L}(g_1 - g_0)\right\|^2 \\
& f_0 \geqslant f_1 + \gamma\langle g_1, g_0\rangle + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\gamma g_0 + \tfrac{1}{L}(g_1 - g_0)\right\|^2 \\
& \|g_0\|^2 = R^2.
\end{aligned}
$$

Semidefinite lifting

- Using $x_1 = x_0 - \gamma g_0$, all elements are quadratic in (g_0, g_1), and linear in (f_0, f_1):

$$\begin{aligned}
\max_{\substack{g_0, g_1 \\ f_0, f_1}} \quad & \|g_1\|^2 \\
\text{subject to} \quad & f_1 \geqslant f_0 - \gamma\|g_0\|^2 + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\gamma g_0 + \tfrac{1}{L}(g_1 - g_0)\right\|^2 \\
& f_0 \geqslant f_1 + \gamma\langle g_1, g_0\rangle + \tfrac{1}{2L}\|g_1 - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|\gamma g_0 + \tfrac{1}{L}(g_1 - g_0)\right\|^2 \\
& \|g_0\|^2 = R^2.
\end{aligned}$$

- They are therefore linear in terms of a Gram matrix G and a vector F, with

$$G = \begin{bmatrix} \|g_0\|^2 & \langle g_0, g_1\rangle \\ \langle g_0, g_1\rangle & \|g_1\|^2 \end{bmatrix} = \begin{bmatrix} g_0 & g_1 \end{bmatrix}^\top \begin{bmatrix} g_0 & g_1 \end{bmatrix}, \quad F = \begin{bmatrix} f_0 & f_1 \end{bmatrix},$$

where $G \succcurlyeq 0$ by construction.

Semidefinite lifting

Semidefinite lifting

◇ Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|g_0\|^2 & \langle g_0, g_1 \rangle \\ \langle g_0, g_1 \rangle & \|g_1\|^2 \end{bmatrix}, \quad F = \begin{bmatrix} f_0 & f_1 \end{bmatrix},$$

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|g_0\|^2 & \langle g_0, g_1 \rangle \\ \langle g_0, g_1 \rangle & \|g_1\|^2 \end{bmatrix}, \quad F = \begin{bmatrix} f_0 & f_1 \end{bmatrix},$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\, F} \quad & G_{2,2} \\ \text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0. \end{aligned}$$

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|g_0\|^2 & \langle g_0, g_1 \rangle \\ \langle g_0, g_1 \rangle & \|g_1\|^2 \end{bmatrix}, \quad F = \begin{bmatrix} f_0 & f_1 \end{bmatrix},$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & G_{2,2} \\ \text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0. \end{aligned}$$

- Assuming $g_0, g_1 \in \mathbb{R}^d$ with $d \geqslant 2$, same optimal value as original problem!

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|g_0\|^2 & \langle g_0, g_1 \rangle \\ \langle g_0, g_1 \rangle & \|g_1\|^2 \end{bmatrix}, \quad F = \begin{bmatrix} f_0 & f_1 \end{bmatrix},$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & G_{2,2} \\ \text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0. \end{aligned}$$

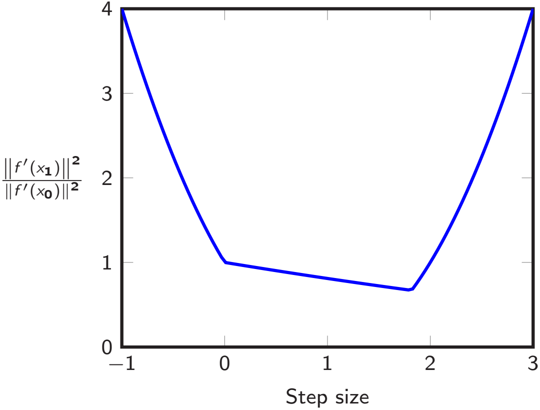
- Assuming $g_0, g_1 \in \mathbb{R}^d$ with $d \geqslant 2$, same optimal value as original problem!
- For $d = 1$ same optimal value by adding $\mathrm{rank}(G) \leqslant 1$.

Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of γ.

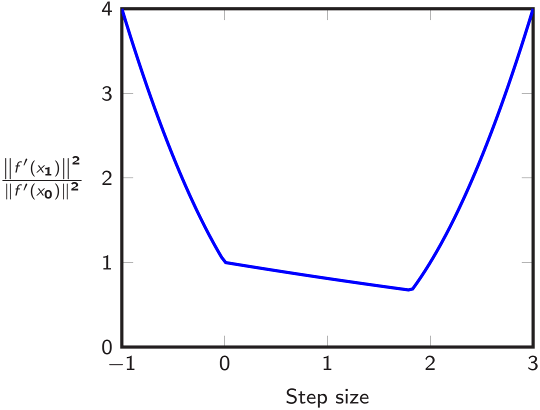
Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of γ.



Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of γ.



Observation: numerics match the (expected) $\max\{(1-\gamma L)^2, (1-\gamma\mu)^2\}$.

Translation to worst-case guarantees

- Let us rephrase our target: we look for $\rho(\gamma)$ (hopefully small) such that

$$\|f'(x_1)\| \leq \rho(\gamma)\|f'(x_0)\|$$

is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - \gamma f'(x_0)$.

Translation to worst-case guarantees

- Let us rephrase our target: we look for $\rho(\gamma)$ (hopefully small) such that

$$\|f'(x_1)\| \leq \rho(\gamma)\|f'(x_0)\|$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - \gamma f'(x_0)$.
- Feasible points to the previous SDP correspond to lower bounds on $\rho(\gamma)$.

Translation to worst-case guarantees

- Let us rephrase our target: we look for $\rho(\gamma)$ (hopefully small) such that

$$\|f'(x_1)\| \leq \rho(\gamma)\|f'(x_0)\|$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - \gamma f'(x_0)$.
- Feasible points to the previous SDP correspond to lower bounds on $\rho(\gamma)$.
- Obtaining upper bounds on $\rho(\gamma)$?

  Exactly what a dual does!

Translation to worst-case guarantees

- Let us rephrase our target: we look for $\rho(\gamma)$ (hopefully small) such that

$$\|f'(x_1)\| \leq \rho(\gamma)\|f'(x_0)\|$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - \gamma f'(x_0)$.
- Feasible points to the previous SDP correspond to lower bounds on $\rho(\gamma)$.
- Obtaining upper bounds on $\rho(\gamma)$?

  Exactly what a dual does!
- Any such $\rho(\gamma)$ that is valid for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$ is a feasible point to the dual SDP.

Dual problem I

Dual problem I

- Recall primal problem

$$
\begin{aligned}
\max_{G,\,F} \quad & G_{2,2} \\
\text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

Dual problem I

- ⋄ Recall primal problem

$$
\begin{aligned}
\max_{G,\,F} \quad & G_{2,2} \\
\text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

- ⋄ Introduce dual variables λ_1, λ_2 and τ for the linear constraints, and dualize.

Dual problem I

- Recall primal problem

$$
\begin{aligned}
\max_{G,\,F} \quad & G_{2,2} \\
\text{subject to} \quad & F_1 - F_0 + \tfrac{\gamma L(2-\gamma\mu)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma\mu}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& F_0 - F_1 + \tfrac{\gamma\mu(2-\gamma L)-1}{2(L-\mu)} G_{1,1} + \tfrac{1-\gamma L}{L-\mu} G_{1,2} - \tfrac{1}{2(L-\mu)} G_{2,2} \geqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

- Introduce dual variables λ_1, λ_2 and τ for the linear constraints, and dualize.
- Dual problem is

$$
\begin{aligned}
\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ & \tau \\
\text{subject to } S = & \begin{bmatrix} -\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1 - \frac{\lambda_1}{L-\mu} \end{bmatrix} \preccurlyeq 0 \\
0 = & \ \lambda_1 - \lambda_2.
\end{aligned}
$$

Dual problem II

- Dual problem is

$$
\begin{aligned}
&\underset{\tau,\lambda_1,\lambda_2 \geqslant 0}{\text{minimize}}\ \tau \\
&\text{subject to } S = \begin{bmatrix} -\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1 - \frac{\lambda_1}{L-\mu} \end{bmatrix} \preccurlyeq 0 \\
&\qquad 0 = \lambda_1 - \lambda_2.
\end{aligned}
$$

Dual problem II

- Dual problem is

$$\begin{aligned}
&\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ \tau\\
&\text{subject to } S=\begin{bmatrix}-\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu}-\tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)}\\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda_1}{L-\mu}\end{bmatrix}\preccurlyeq 0\\
&\qquad\qquad 0=\lambda_1-\lambda_2.
\end{aligned}$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate

Dual problem II

- Dual problem is

$$\begin{aligned}
&\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ \tau\\
&\text{subject to } S=\begin{bmatrix}-\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu}-\tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)}\\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda_1}{L-\mu}\end{bmatrix}\preccurlyeq 0\\
&\qquad 0=\lambda_1-\lambda_2.
\end{aligned}$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate
- Direct consequence:

$$\|f'(x_1)\|^2\leq\tau\|f'(x_0)\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d \text{ and all } d\in\mathbb{N}$$

$$\Uparrow$$

$$\exists\lambda\geq 0: \begin{bmatrix}-\frac{\lambda(\gamma\mu-1)(\gamma L-1)}{L-\mu}-\tau & -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)}\\ -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda}{L-\mu}\end{bmatrix}\preccurlyeq 0$$

Dual problem II

- Dual problem is

$$
\begin{aligned}
\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ & \tau \\
\text{subject to } & S = \begin{bmatrix} -\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda_1}{L-\mu} \end{bmatrix} \preccurlyeq 0 \\
& 0 = \lambda_1 - \lambda_2.
\end{aligned}
$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$)
- Direct consequence:

$$
\|f'(x_1)\|^2 \leq \tau\|f'(x_0)\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d \text{ and all } d\in\mathbb{N}
$$

$$
\Uparrow
$$

$$
\exists\lambda\geq 0: \begin{bmatrix} -\frac{\lambda(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda}{L-\mu} \end{bmatrix} \preccurlyeq 0
$$

Dual problem II

- Dual problem is

$$\begin{aligned}
&\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ \tau\\
&\text{subject to } S=\begin{bmatrix}-\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu}-\tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)}\\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda_1}{L-\mu}\end{bmatrix}\preccurlyeq 0\\
&\qquad 0=\lambda_1-\lambda_2.
\end{aligned}$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$)
- Direct consequence:

$$\boxed{\begin{gathered}\|f'(x_1)\|^2\leq\tau\|f'(x_0)\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d \text{ and all } d\in\mathbb{N}\\ \Uparrow\\ \exists\lambda\geq 0:\ \begin{bmatrix}-\frac{\lambda(\gamma\mu-1)(\gamma L-1)}{L-\mu}-\tau & -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)}\\ -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda}{L-\mu}\end{bmatrix}\preccurlyeq 0\end{gathered}}$$

- Strong duality holds (existence of a Slater point): any valid worst-case convergence rate $\equiv$ valid dual feasible point ($\Downarrow$) .

Dual problem II

- Dual problem is

$$\begin{aligned} &\underset{\tau,\lambda_1,\lambda_2\geqslant 0}{\text{minimize}}\ \tau \\ &\text{subject to } S = \begin{bmatrix} -\frac{\lambda_1(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda_1(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda_1}{L-\mu} \end{bmatrix} \preccurlyeq 0 \\ &\qquad 0 = \lambda_1 - \lambda_2. \end{aligned}$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$)
- Direct consequence:

$$\|f'(x_1)\|^2 \leq \tau\|f'(x_0)\|^2 \text{ for all } f \in \mathcal{F}_{\mu,L}, \text{ all } x_0 \in \mathbb{R}^d \text{ and all } d \in \mathbb{N}$$

$$\Updownarrow$$

$$\exists \lambda \geq 0 : \begin{bmatrix} -\frac{\lambda(\gamma\mu-1)(\gamma L-1)}{L-\mu} - \tau & -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} \\ -\frac{\lambda(\gamma(\mu+L)-2)}{2(L-\mu)} & 1-\frac{\lambda}{L-\mu} \end{bmatrix} \preccurlyeq 0$$

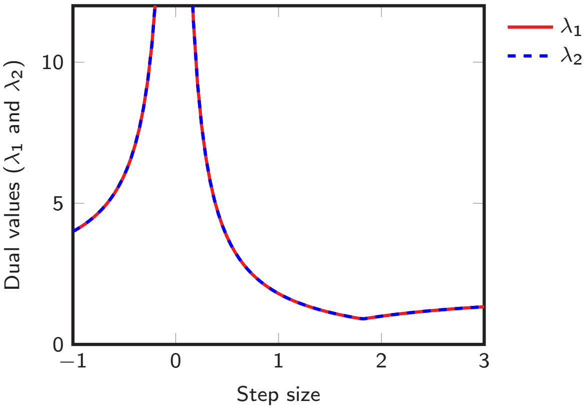
- Strong duality holds (existence of a Slater point): any valid worst-case convergence rate $\equiv$ valid dual feasible point ($\Downarrow$) : hence "$\Updownarrow$".

Solving the dual

Fix $L = 1$, $\mu = .1$ and solve the dual SDP for a few values of γ.

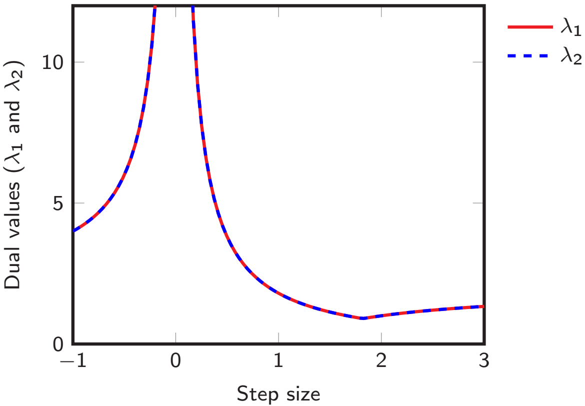
Solving the dual

Fix $L = 1$, $\mu = .1$ and solve the dual SDP for a few values of γ.



Solving the dual

Fix $L = 1$, $\mu = .1$ and solve the dual SDP for a few values of γ.



Note: numerics match $\lambda_1 = \lambda_2 = \frac{2}{|\gamma|}\rho(\gamma)$ with $\rho(\gamma) = \max\{|1 - \gamma L|, |1 - \gamma\mu|\}$.

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$
\begin{aligned}
f_0 \geqslant f_1 \quad & + \langle f'(x_1), x_0 - x_1 \rangle + \tfrac{1}{2L} \| f'(x_0) - f'(x_1) \|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 && : \lambda_1 \\
f_1 \geqslant f_0 \quad & + \langle f'(x_0), x_1 - x_0 \rangle + \tfrac{1}{2L} \| f'(x_0) - f'(x_1) \|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 && : \lambda_2
\end{aligned}
$$

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$
\begin{array}{lll}
f_0 \geqslant f_1 & +\langle f'(x_1), x_0 - x_1\rangle + \frac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 & \\
& +\frac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \frac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_1 \\
f_1 \geqslant f_0 & +\langle f'(x_0), x_1 - x_0\rangle + \frac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 & \\
& +\frac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \frac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_2
\end{array}
$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$
\begin{aligned}
f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1 \rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0 \rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}
$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$\begin{aligned} f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\ f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma) \end{aligned}$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$\left\|f'(x_1)\right\|^2 \leqslant (1-\gamma\mu)^2\left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{},$$

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$\begin{aligned} f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma)$$

$$\begin{aligned} f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$\left\|f'(x_1)\right\|^2 \leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geqslant 0},$$

Recovering a “standard” proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$\begin{aligned} f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_1 = \frac{2}{\gamma}(1-\mu\gamma)$$

$$\begin{aligned} f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_2 = \frac{2}{\gamma}(1-\mu\gamma)$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$\begin{aligned} \left\|f'(x_1)\right\|^2 & \leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geqslant 0}, \\ & \leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2, \end{aligned}$$

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$
\begin{aligned}
f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}
$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$
\begin{aligned}
\left\|f'(x_1)\right\|^2 &\leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma) f'(x_0) - f'(x_1)\right\|^2}_{\geqslant 0}, \\
&\leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2,
\end{aligned}
$$

leading to $\|f'(x_1)\|^2 \leqslant (1-\frac{\mu}{L})^2 \|f'(x_0)\|^2$

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$\begin{aligned}
f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 & \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 & \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$\begin{aligned}
\left\|f'(x_1)\right\|^2 &\leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geqslant 0,\ \text{or } = 0 \text{ when worst-case is achieved}}, \\
&\leqslant (1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2,
\end{aligned}$$

leading to $\|f'(x_1)\|^2 \leqslant (1 - \frac{\mu}{L})^2 \|f'(x_0)\|^2$

Recovering a "standard" proof

Gradient with $\gamma = \frac{1}{L}$. Perform weighted sum of two inequalities

$$\begin{aligned} f_0 \geqslant f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma)$$

$$\begin{aligned} f_1 \geqslant f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \end{aligned} \qquad : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)$$

with $\lambda_1, \lambda_2 \geqslant 0$. Weighted sum can be reformulated as

$$\begin{aligned} \left\|f'(x_1)\right\|^2 &\leqslant (1-\gamma\mu)^2\left\|f'(x_0)\right\|^2 - \underbrace{\tfrac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geqslant 0, \text{ or } = 0 \text{ when worst-case is achieved}}, \\ &\leqslant (1-\gamma\mu)^2\left\|f'(x_0)\right\|^2, \end{aligned}$$

leading to $\|f'(x_1)\|^2 \leqslant (1-\frac{\mu}{L})^2\|f'(x_0)\|^2$ (tight).

Remarks

Dual interpretations:

Remarks

Dual interpretations:

- Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.

Remarks

Dual interpretations:

- ◇ Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.
- ◇ From strong duality: in such settings, any (dimension-independent) convergence rate can be proved by linear combination of interpolation inequalities.

Remarks

Dual interpretations:

- ◇ Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.
- ◇ From strong duality: in such settings, any (dimension-independent) convergence rate can be proved by linear combination of interpolation inequalities.
- ◇ Any dual feasible point can be translated into a "traditional" (SDP-less) proof.

Remarks

Dual interpretations:

- ◇ Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.
- ◇ From strong duality: in such settings, any (dimension-independent) convergence rate can be proved by linear combination of interpolation inequalities.
- ◇ Any dual feasible point can be translated into a "traditional" (SDP-less) proof.

For finding proofs:

Remarks

Dual interpretations:

- ◇ Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.
- ◇ From strong duality: in such settings, any (dimension-independent) convergence rate can be proved by linear combination of interpolation inequalities.
- ◇ Any dual feasible point can be translated into a "traditional" (SDP-less) proof.

For finding proofs:

- ◇ the SDP might help by playing with both sides:
  - – play with primal (e.g., worst-case functions might be easy to identify),
  - – play with dual (e.g., dual variables might be easy to identify).

Remarks

Dual interpretations:

- ◇ Find smallest convergence rate that can be proved by a linear combination of interpolation inequalities.
- ◇ From strong duality: in such settings, any (dimension-independent) convergence rate can be proved by linear combination of interpolation inequalities.
- ◇ Any dual feasible point can be translated into a "traditional" (SDP-less) proof.

For finding proofs:

- ◇ the SDP might help by playing with both sides:
  - play with primal (e.g., worst-case functions might be easy to identify),
  - play with dual (e.g., dual variables might be easy to identify).
- ◇ Standard tricks apply, e.g., trace norm minimization for promoting low-rank solutions (on primal or dual).

When does it work?

Problem setting:

- ◇ pick a method
- ◇ pick a class of functions
- ◇ pick a type of inequality we want to reach

  (e.g., via a convergence measure & an initial condition).

When does it work?

Problem setting:

- $\diamond$ pick a method
- $\diamond$ pick a class of functions
- $\diamond$ pick a type of inequality we want to reach

  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

When does it work?

Problem setting:

- ◇ pick a method
- ◇ pick a class of functions
- ◇ pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- ◇ Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).

When does it work?

Problem setting:

- ◇ pick a method
- ◇ pick a class of functions
- ◇ pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- ◇ Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).
- ◇ Class of function $\mathcal{F}_{\mu,\mathrm{L}}$ was encoded via linear constraints in G and F.

When does it work?

Problem setting:

- ⋄ pick a method
- ⋄ pick a class of functions
- ⋄ pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- ⋄ Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).
- ⋄ Class of function $\mathcal{F}_{\mu,L}$ was encoded via linear constraints in G and F.
- ⋄ Performance measure $\|f'(x_1)\|^2$ was linear in terms of G and F.

When does it work?

Problem setting:

- pick a method
- pick a class of functions
- pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).
- Class of function $\mathcal{F}_{\mu,\mathrm{L}}$ was encoded via linear constraints in G and F.
- Performance measure $\|f'(x_1)\|^2$ was linear in terms of G and F.
- Initial condition $\|f'(x_0)\|^2$ was linear in terms of G and F.

When does it work?

Problem setting:

- ⋄ pick a method
- ⋄ pick a class of functions
- ⋄ pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- ⋄ Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).
- ⋄ Class of function $\mathcal{F}_{\mu,\mathrm{L}}$ was encoded via linear constraints in G and F.
- ⋄ Performance measure $\|f'(x_1)\|^2$ was linear in terms of G and F.
- ⋄ Initial condition $\|f'(x_0)\|^2$ was linear in terms of G and F.

... such conditions (or slight generalizations) hold in variety of cases (see later).

When does it work?

Problem setting:

- ◇ pick a method
- ◇ pick a class of functions
- ◇ pick a type of inequality we want to reach
  (e.g., via a convergence measure & an initial condition).

Why could we solve the previous PEP?

- ◇ Step size γ was "fixed beforehand"; no dependence on $f(.)$ (non-adaptive).
- ◇ Class of function $\mathcal{F}_{\mu,L}$ was encoded via linear constraints in G and F.
- ◇ Performance measure $\|f'(x_1)\|^2$ was linear in terms of G and F.
- ◇ Initial condition $\|f'(x_0)\|^2$ was linear in terms of G and F.

... such conditions (or slight generalizations) hold in variety of cases (see later).

In other situations, one might want to relax the PEP for obtaining upper-bounds.

Going further

Going further

- Sublinear rates?

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

    $$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- Lyapunov functions?

Going further

- ◇ Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- ◇ Lyapunov functions?
  - Example: let $V_k = a\|x_k - x_\star\|^2 + b\|f'(x_k)\|^2 + c(f(x_k) - f_\star)$.

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- Lyapunov functions?
  - Example: let $V_k = a\|x_k - x_\star\|^2 + b\|f'(x_k)\|^2 + c(f(x_k) - f_\star)$.
  - For fixed ρ, feasibility problem

$$\text{“}?\exists a, b, c \;\text{ s.t. }\; V_{k+1} \leqslant \rho V_k\text{”}$$

    is convex (to convince yourself, just write the dual PEP: linear in a, b, c).

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

$$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- Lyapunov functions?
  - Example: let $V_k = a\|x_k - x_\star\|^2 + b\|f'(x_k)\|^2 + c(f(x_k) - f_\star)$.
  - For fixed ρ, feasibility problem

$$\text{“}?\exists a, b, c \text{ s.t. } V_{k+1} \leqslant \rho V_k\text{”}$$

    is convex (to convince yourself, just write the dual PEP: linear in a, b, c).
  - Minimization over ρ is quasi-convex (bilinear in a, b, c and ρ) solved via bisection method + SDP solver.

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

    $$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- Lyapunov functions?
  - Example: let $V_k = a\|x_k - x_\star\|^2 + b\|f'(x_k)\|^2 + c(f(x_k) - f_\star)$.
  - For fixed ρ, feasibility problem

    $$\text{“?}\exists a, b, c \;\text{ s.t. }\; V_{k+1} \leqslant \rho V_k\text{”}$$

    is convex (to convince yourself, just write the dual PEP: linear in a, b, c).
  - Minimization over ρ is quasi-convex (bilinear in a, b, c and ρ) solved via bisection method + SDP solver.
  - Similar to “integral quadratic constraints” by Lessard et al. (2016).

Going further

- Sublinear rates?
  - Look for different types of guarantees, for example

    $$f(x_N) - f(x_\star) \leqslant C_N \|x_0 - x_\star\|^2,$$

    for some C_N (hopefully small and decreasing with N).
  - Similar ideas and larger SDPs (typically of order $N \times N$),
  - and $O(N^2)$ interpolation constraints.

- Lyapunov functions?
  - Example: let $V_k = a\|x_k - x_\star\|^2 + b\|f'(x_k)\|^2 + c(f(x_k) - f_\star)$.
  - For fixed ρ, feasibility problem

    $$\text{"?}\exists a, b, c \ \text{ s.t. } \ V_{k+1} \leqslant \rho V_k\text{"}$$

    is convex (to convince yourself, just write the dual PEP: linear in a, b, c).
  - Minimization over ρ is quasi-convex (bilinear in a, b, c and ρ) solved via bisection method + SDP solver.
  - Similar to "integral quadratic constraints" by Lessard et al. (2016).

- Optimizing/designing methods? upcoming!

Avoiding semidefinite programming modeling steps?

Avoiding semidefinite programming modeling steps?

François Glineur
(UCLouvain)

Julien Hendrickx
(UCLouvain)

"Performance Estimation Toolbox (PESTO): automated worst-case analysis of first-order optimization methods" (CDC 2017)

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L} f'(y_i)$$
$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L} f'(y_i)$$
$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{d}_f(y_i) - f'(y_i)\| \leq \varepsilon \|f'(y_i)\|.$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L} f'(y_i)$$

$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{\mathbf{d}}_{\mathbf{f}}(\mathbf{y}_{\mathbf{i}}) - f'(y_i)\| \leq \varepsilon \|f'(y_i)\|.$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L}\tilde{\mathbf{d}}_{\mathbf{f}}(\mathbf{y}_{\mathbf{i}})$$

$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{\mathbf{d}}_{\mathbf{f}}(\mathbf{y}_{\mathbf{i}}) - f'(y_i)\| \leq \varepsilon \|f'(y_i)\|.$$

PESTO example: an inexact fast gradient method

```matlab
% (0) Initialize an empty PEP
P = pep();

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0       = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();         % xs is an optimal point, and fs=F(xs)
P.InitialCondition((x0-xs)^2 <= 1);  % Add an initial condition ||x0-xs||^2<= 1

% (3) Algorithm
N = 7; % number of iterations

x    = cell(N+1,1); % we store the iterates in a cell for convenience
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```

PESTO example: an inexact fast gradient method

```
% (0) Initialize an empty PEP
P = pep();

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0       = P.StartingPoint();       % x0 is some starting point
[xs, fs] = F.OptimalPoint();        % xs is an optimal point, and fs=F(xs)
```

```
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```

```
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```

PESTO example: an inexact fast gradient method

```
% (0) Initialize an empty PEP
P = pep();

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0      = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();        % xs is an optimal point, and fs=F(xs)
```

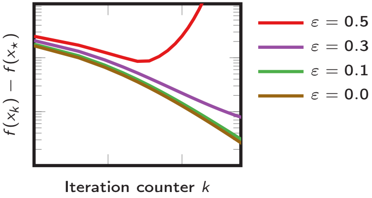
```
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```


```
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```


```
% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});     % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```



PESTO example: Douglas-Rachford splitting

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}     = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation

for k = 1 : N
    x{k}    = proximal_step(w{k},B,lambda);
    y{k}    = proximal_step(2*x{k}-w{k},A,lambda);
    w{k+1}  = w{k}-theta*(x{k}-y{k});

    xp{k}   = proximal_step(wp{k},B,lambda);
    yp{k}   = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}   = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)   % worst-case contraction factor
```

PESTO example: Douglas-Rachford splitting

```matlab
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}     = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation

x{k}     = proximal_step(w{k},B,lambda);
y{k}     = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}   = w{k}-theta*(x{k}-y{k});
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)   % worst-case contraction factor
```

PESTO example: Douglas-Rachford splitting

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}    = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation
```

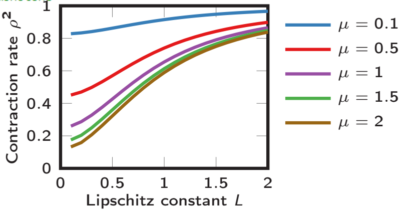
```
x{k}    = proximal_step(w{k},B,lambda);
y{k}    = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}  = w{k}-theta*(x{k}-y{k});
```

```
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)   % worst-case contraction factor
```



PESTO example: Douglas-Rachford splitting

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}    = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation
```

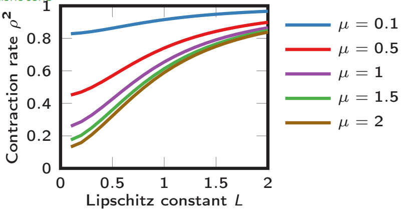
```
x{k}     = proximal_step(w{k},B,lambda);
y{k}     = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}   = w{k}-theta*(x{k}-y{k});
```

```
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)   % worst-case contraction factor
```



✔ fast prototyping ($\sim$ 20 effective lines)
✔ quick analyses ($\sim$ 10 minutes)
✔ computer-aided proofs (multipliers)

Current library of examples within PESTO

Includes... but not limited to

- ◇ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ◇ proximal point algorithm,
- ◇ projected and proximal gradient, accelerated/momentum versions,
- ◇ steepest descent, greedy/conjugate gradient methods,
- ◇ Douglas-Rachford/three operator splitting,
- ◇ Frank-Wolfe/conditional gradient,
- ◇ inexact gradient/fast gradient,
- ◇ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ◇ mirror descent,
- ◇ stochastic methods: SAG, SAGA, SGD and variants.

Current library of examples within PESTO

Includes... but not limited to

- ◇ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ◇ proximal point algorithm,
- ◇ projected and proximal gradient, accelerated/momentum versions,
- ◇ steepest descent, greedy/conjugate gradient methods,
- ◇ Douglas-Rachford/three operator splitting,
- ◇ Frank-Wolfe/conditional gradient,
- ◇ inexact gradient/fast gradient,
- ◇ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ◇ mirror descent,
- ◇ stochastic methods: SAG, SAGA, SGD and variants.

PESTO contains most of the recent PEP-related advances (including techniques by other groups). Clean updated references in user manual.

Current library of examples within PESTO

Includes... but not limited to

- subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- proximal point algorithm,
- projected and proximal gradient, accelerated/momentum versions,
- steepest descent, greedy/conjugate gradient methods,
- Douglas-Rachford/three operator splitting,
- Frank-Wolfe/conditional gradient,
- inexact gradient/fast gradient,
- Krasnoselskii-Mann and Halpern fixed-point iterations,
- mirror descent,
- stochastic methods: SAG, SAGA, SGD and variants.

PESTO contains most of the recent PEP-related advances (including techniques by other groups). Clean updated references in user manual.

Among others, see works by Drori, Teboulle, Kim, Fessler, Ryu, Lieder, Lessard, Recht, Packard, Van Scoy, Cyrus, Gu, Yang, etc.

Current library of examples within PESTO

Includes... but not limited to

- ⋄ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ⋄ proximal point algorithm,
- ⋄ projected and proximal gradient, accelerated/momentum versions,
- ⋄ steepest descent, greedy/conjugate gradient methods,
- ⋄ Douglas-Rachford/three operator splitting,
- ⋄ Frank-Wolfe/conditional gradient,
- ⋄ inexact gradient/fast gradient,
- ⋄ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ⋄ mirror descent,
- ⋄ stochastic methods: SAG, SAGA, SGD and variants.

PESTO contains most of the recent PEP-related advances (including techniques by other groups). Clean updated references in user manual.

Among others, see works by Drori, Teboulle, Kim, Fessler, Ryu, Lieder, Lessard, Recht, Packard, Van Scoy, Cyrus, Gu, Yang, etc.

Currently in Matlab, soon in Python.

Performance estimation problems

Designing methods using PEPs

Conclusions

Main inspiration

Great inspiration from previous works.

- B. Polyak. "Introduction to optimization" (1964)
- Y. Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". (1983)
- A. Nemirovsky, and B. Polyak. "Iterative methods for solving linear ill-posed problems under precise information." (1984)
- A. Nemirovsky. "Information-based complexity of linear operator equations". (1992)
- A. Nemirovsky. "Information-based complexity of convex programming". (lecture notes, 1995)
- Y. Nesterov. "Introductory Lectures on Convex Optimization". (2003/2018)

and many others.

Main inspiration

Great inspiration from previous works.

- ◇ B. Polyak. "Introduction to optimization" (1964)
- ◇ Y. Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". (1983)
- ◇ A. Nemirovsky, and B. Polyak. "Iterative methods for solving linear ill-posed problems under precise information." (1984)
- ◇ A. Nemirovsky. "Information-based complexity of linear operator equations". (1992)
- ◇ A. Nemirovsky. "Information-based complexity of convex programming". (lecture notes, 1995)
- ◇ Y. Nesterov. "Introductory Lectures on Convex Optimization". (2003/2018)

and many others.

In next couple of slides:

- ◇ goal: principled way towards optimal methods.
- ◇ in some sense, *generalization* of Chebyshev methods (tailored for quadratic minimization) to non-quadratic smooth strongly convex setup.

Main references

Yoel Drori

Main references

Yoel Drori

Main references for the following slides (sloppy references throughout):

- ◇ Y. Drori, T., "On the oracle complexity of smooth strongly convex minimization". (2021)
- ◇ T., Y. Drori, "An optimal gradient method for smooth strongly convex minimization". (2021)
- ◇ Y. Drori, T., "Efficient first-order methods for convex minimization: a constructive approach". (2020)

Main references

Yoel Drori

Main references for the following slides (sloppy references throughout):

- ⋄ Y. Drori, T., "On the oracle complexity of smooth strongly convex minimization". (2021)
- ⋄ T., Y. Drori, "An optimal gradient method for smooth strongly convex minimization". (2021)
- ⋄ Y. Drori, T., "Efficient first-order methods for convex minimization: a constructive approach". (2020)
- ⋄ Y. Drori, "The exact information-based complexity of smooth convex minimization". (2017)
- ⋄ D. Kim, J.F. Fessler, "Optimized first-order methods for smooth convex minimization". (2016)
- ⋄ Y. Drori, M. Teboulle, "Performance of first-order methods for smooth convex minimization: a novel approach". (2014)

Main references

Yoel Drori

Main references for the following slides (sloppy references throughout):

- ⋄ Y. Drori, T., "On the oracle complexity of smooth strongly convex minimization". (2021)
- ⋄ T., Y. Drori, "An optimal gradient method for smooth strongly convex minimization". (2021)
- ⋄ Y. Drori, T., "Efficient first-order methods for convex minimization: a constructive approach". (2020)
- ⋄ Y. Drori, "The exact information-based complexity of smooth convex minimization". (2017)
- ⋄ D. Kim, J.F. Fessler, "Optimized first-order methods for smooth convex minimization". (2016)
- ⋄ Y. Drori, M. Teboulle, "Performance of first-order methods for smooth convex minimization: a novel approach". (2014)

Also closely related:

- ⋄ B. Van Scoy, R.A. Freeman, K.M. Lynch, "The fastest known globally convergent first-order method for minimizing strongly convex functions". (2017)
- ⋄ D. Kim, J.F. Fessler, "Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions". (2021)

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- ◇ Upper bound side: PEP is a machinery for designing worst-case guarantees

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- Upper bound side: PEP is a machinery for designing worst-case guarantees
  - idea 1: what about optimizing worst-case guarantees?
    (design via minimax problems)

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- Upper bound side: PEP is a machinery for designing worst-case guarantees
  - idea 1: what about optimizing worst-case guarantees?
    (design via minimax problems)
  - idea 2: what about "mimicking" an ideal method?
    (design via conjugate gradients, but no time today)

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- Upper bound side: PEP is a machinery for designing worst-case guarantees
  - idea 1: what about optimizing worst-case guarantees?
    (design via minimax problems)
  - idea 2: what about "mimicking" an ideal method?
    (design via conjugate gradients, but no time today)
- Lower bound side: interpolation/extension theorems

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- ◇ Upper bound side: PEP is a machinery for designing worst-case guarantees
  - idea 1: what about optimizing worst-case guarantees?
    (design via minimax problems)
  - idea 2: what about "mimicking" an ideal method?
    (design via conjugate gradients, but no time today)
- ◇ Lower bound side: interpolation/extension theorems
  - provide a constructive way to generate worst-case examples,

Designing methods using PEPs

How could we use PEPs for designing methods, and lower complexity bounds?

- ⋄ Upper bound side: PEP is a machinery for designing worst-case guarantees
  - idea 1: what about optimizing worst-case guarantees?
    (design via minimax problems)
  - idea 2: what about "mimicking" an ideal method?
    (design via conjugate gradients, but no time today)
- ⋄ Lower bound side: interpolation/extension theorems
  - provide a constructive way to generate worst-case examples,
  - can be used for designing "worst functions in the world".

Design via minimax problems

We need a few ingredients:

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

(ii) A class of methods.

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

(ii) A class of methods. For example

$$\begin{aligned}
w_1 &= w_0 - h_{1,0} f'(w_0),\\
w_2 &= w_1 - h_{2,0} f'(w_0) - h_{2,1} f'(w_1),\\
w_3 &= w_2 - h_{3,0} f'(w_0) - h_{3,1} f'(w_1) - h_{3,2} f'(w_2),\\
&\;\vdots\\
w_N &= w_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(w_i).
\end{aligned} \tag{FOM}$$

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

(ii) A class of methods. For example

$$\begin{aligned}
w_1 &= w_0 - h_{1,0} f'(w_0), \\
w_2 &= w_1 - h_{2,0} f'(w_0) - h_{2,1} f'(w_1), \\
w_3 &= w_2 - h_{3,0} f'(w_0) - h_{3,1} f'(w_1) - h_{3,2} f'(w_2), \\
&\vdots \\
w_N &= w_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(w_i).
\end{aligned} \tag{FOM}$$

(iii) A notion of "accuracy".

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

(ii) A class of methods. For example

$$\begin{aligned}
w_1 &= w_0 - h_{1,0} f'(w_0),\\
w_2 &= w_1 - h_{2,0} f'(w_0) - h_{2,1} f'(w_1),\\
w_3 &= w_2 - h_{3,0} f'(w_0) - h_{3,1} f'(w_1) - h_{3,2} f'(w_2),\\
&\;\;\vdots\\
w_N &= w_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(w_i).
\end{aligned} \tag{FOM}$$

(iii) A notion of "accuracy". Examples: $\frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2}$, $\frac{\|w_N-w_\star\|^2}{\|w_0-w_\star\|^2}$, $\frac{\|f'(w_N)\|^2}{f(w_0)-f_\star}$.

Design via minimax problems

We need a few ingredients:

(i) a class of problems. Here: L-smooth μ-strongly convex functions $\mathcal{F}_{\mu,L}$

(ii) A class of methods. For example

$$\begin{aligned}
w_1 &= w_0 - h_{1,0} f'(w_0),\\
w_2 &= w_1 - h_{2,0} f'(w_0) - h_{2,1} f'(w_1),\\
w_3 &= w_2 - h_{3,0} f'(w_0) - h_{3,1} f'(w_1) - h_{3,2} f'(w_2),\\
&\vdots\\
w_N &= w_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(w_i).
\end{aligned} \tag{FOM}$$

(iii) A notion of "accuracy". Examples: $\frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2}$, $\frac{\|w_N-w_\star\|^2}{\|w_0-w_\star\|^2}$, $\frac{\|f'(w_N)\|^2}{f(w_0)-f_\star}$.

$\Rightarrow$ Resulting design problem, for example

$$\min_{\{h_{i,j}\}} \max_{f\in\mathcal{F}_{0,L}} \left\{ \frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2} \,:\, w_N \text{ obtained from (FOM) and } w_0 \right\}.$$

(i.e., "minimize worst-case")

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- results are typically obtained via rescaled/shifted Chebyshev polynomials.

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- ◇ results are typically obtained via rescaled/shifted Chebyshev polynomials.
- ◇ Optimal methods are known for a few notions of accuracy, including

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- ◇ results are typically obtained via rescaled/shifted Chebyshev polynomials.
- ◇ Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- results are typically obtained via rescaled/shifted Chebyshev polynomials.
- Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex quadratics).

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- ◇ results are typically obtained via rescaled/shifted Chebyshev polynomials.
- ◇ Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex quadratics).
- ◇ Those methods have convenient formulations (no need to memorize all past coefficients and gradients)

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- results are typically obtained via rescaled/shifted Chebyshev polynomials.
- Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex quadratics).
- Those methods have convenient formulations (no need to memorize all past coefficients and gradients)
  - Ex: for $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$, we get "Chebyshev semi-iterative method":

$$w_k = w_{k-1} - \frac{4\delta_k}{L-\mu} f'(w_{k-1}) + \left(1 - \frac{2\delta_k(L+\mu)}{L-\mu}\right)(w_{k-2} - w_{k-1}),$$

with $\delta_k := \left(2\frac{L+\mu}{L-\mu} - \delta_{k-1}\right)^{-1}$ and $\delta_0 := \frac{L+\mu}{L-\mu}$.

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- results are typically obtained via rescaled/shifted Chebyshev polynomials.
- Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex quadratics).
- Those methods have convenient formulations (no need to memorize all past coefficients and gradients)
  - Ex: for $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$, we get "Chebyshev semi-iterative method":

$$w_k = w_{k-1} - \frac{4\delta_k}{L-\mu} f'(w_{k-1}) + \left(1 - \frac{2\delta_k(L+\mu)}{L-\mu}\right)(w_{k-2} - w_{k-1}),$$

with $\delta_k := \left(2\frac{L+\mu}{L-\mu} - \delta_{k-1}\right)^{-1}$ and $\delta_0 := \frac{L+\mu}{L-\mu}$. Limit case (as $k \to \infty$) is Polyak's Heavy-ball method.

Relation to Chebyshev methods

When f is a quadratic function, such design procedures are known

- ◇ results are typically obtained via rescaled/shifted Chebyshev polynomials.
- ◇ Optimal methods are known for a few notions of accuracy, including
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex quadratics),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex quadratics).
- ◇ Those methods have convenient formulations (no need to memorize all past coefficients and gradients)
  - Ex: for $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$, we get "Chebyshev semi-iterative method":

$$w_k = w_{k-1} - \frac{4\delta_k}{L-\mu} f'(w_{k-1}) + \left(1 - \frac{2\delta_k(L+\mu)}{L-\mu}\right)(w_{k-2} - w_{k-1}),$$

    with $\delta_k := \left(2\frac{L+\mu}{L-\mu} - \delta_{k-1}\right)^{-1}$ and $\delta_0 := \frac{L+\mu}{L-\mu}$. Limit case (as $k \to \infty$) is Polyak's Heavy-ball method.
- ◇ The situation actually quite similar beyond quadratics.

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,

Minimax design beyond quadratics

From my humble current understanding:

- ◇ approaches to minimax (beyond quadratics) are less direct,
- ◇ traditionally follows a two-stage procedure:

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
    - (i) algorithm-dependent upper bound, and

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
   (i) algorithm-dependent upper bound, and
   (ii) algorithm-independent lower bound.

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
  - (i) algorithm-dependent upper bound, and
  - (ii) algorithm-independent lower bound.
- Similar approach with PEPs, helped by computers and SDPs.

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
  (i) algorithm-dependent upper bound, and
  (ii) algorithm-independent lower bound.
- Similar approach with PEPs, helped by computers and SDPs.
- It turns out that optimal methods are now known for the two same notions

Minimax design beyond quadratics

From my humble current understanding:

- ◇ approaches to minimax (beyond quadratics) are less direct,
- ◇ traditionally follows a two-stage procedure:
  - (i) algorithm-dependent upper bound, and
  - (ii) algorithm-independent lower bound.
- ◇ Similar approach with PEPs, helped by computers and SDPs.
- ◇ It turns out that optimal methods are now known for the two same notions
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex functions),

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
  (i) algorithm-dependent upper bound, and
  (ii) algorithm-independent lower bound.
- Similar approach with PEPs, helped by computers and SDPs.
- It turns out that optimal methods are now known for the two same notions
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex functions),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex functions),

Minimax design beyond quadratics

From my humble current understanding:

- approaches to minimax (beyond quadratics) are less direct,
- traditionally follows a two-stage procedure:
  - (i) algorithm-dependent upper bound, and
  - (ii) algorithm-independent lower bound.
- Similar approach with PEPs, helped by computers and SDPs.
- It turns out that optimal methods are now known for the two same notions
  - $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex functions),
  - $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex functions),
  - those methods are incredibly close to Nesterov's method!

Minimax design beyond quadratics

From my humble current understanding:

- ⋄ approaches to minimax (beyond quadratics) are less direct,
- ⋄ traditionally follows a two-stage procedure:
  - (i) algorithm-dependent upper bound, and
  - (ii) algorithm-independent lower bound.
- ⋄ Similar approach with PEPs, helped by computers and SDPs.
- ⋄ It turns out that optimal methods are now known for the two same notions
  - – $\frac{\|w_N - w_\star\|^2}{\|w_0 - w_\star\|^2}$ (for L-smooth μ-strongly convex functions),
  - – $\frac{f(w_N) - f_\star}{\|w_0 - w_\star\|^2}$ (for L-smooth convex functions),
  - – those methods are incredibly close to Nesterov's method!
- ⋄ Beyond that, a few criterion/settings/methods for which "perfectly optimal" algorithms might be known, but matching lower bounds are still missing.
  - – $\frac{\|f'(w_N)\|^2}{f(w_0) - f_\star}$,
  - – a few numerically-generated methods.

Optimized gradient method (OGM)

Example I: the optimal method for $\frac{f(y_N)-f_\star}{\|y_0-y_\star\|^2}$ is called the "optimized gradient method" (OGM, Kim & Fessler (2016)):

Optimized gradient method (OGM)

Example I: the optimal method for $\frac{f(y_N)-f_\star}{\|y_0-y_\star\|^2}$ is called the "optimized gradient method" (OGM, Kim & Fessler (2016)):

$$
\begin{aligned}
y_k &= \frac{1}{\theta_{k,N}} z_k + \left(1 - \frac{1}{\theta_{k,N}}\right)\left(y_{k-1} - \frac{1}{L} f'(y_{k-1})\right) \\
z_{k+1} &= z_k - \frac{2\theta_{i,N}}{L} f'(y_k)
\end{aligned}
$$

Optimized gradient method (OGM)

Example I: the optimal method for $\frac{f(y_N)-f_\star}{\|y_0-y_\star\|^2}$ is called the "optimized gradient method" (OGM, Kim & Fessler (2016)):

$$
\begin{aligned}
y_k &= \frac{1}{\theta_{k,N}} z_k + \left(1 - \frac{1}{\theta_{k,N}}\right) \left(y_{k-1} - \tfrac{1}{L} f'(y_{k-1})\right) \\
z_{k+1} &= z_k - \frac{2\theta_{i,N}}{L} f'(y_k)
\end{aligned}
$$

which rely on some exotic sequence

$$
\theta_{k+1,N} = \begin{cases} \frac{1+\sqrt{4\theta_{k,N}^2+1}}{2} & \text{if } k \leqslant N-2 \\ \frac{1+\sqrt{8\theta_{k,N}^2+1}}{2} & \text{if } k = N-1, \end{cases}
$$

Optimized gradient method (OGM)

Example I: the optimal method for $\frac{f(y_N)-f_\star}{\|y_0-y_\star\|^2}$ is called the "optimized gradient method" (OGM, Kim & Fessler (2016)):

$$\begin{aligned} y_k &= \frac{1}{\theta_{k,N}} z_k + \left(1 - \frac{1}{\theta_{k,N}}\right)\left(y_{k-1} - \tfrac{1}{L} f'(y_{k-1})\right) \\ z_{k+1} &= z_k - \frac{2\theta_{i,N}}{L} f'(y_k) \end{aligned}$$

which rely on some exotic sequence

$$\theta_{k+1,N} = \begin{cases} \frac{1+\sqrt{4\theta_{k,N}^2+1}}{2} & \text{if } k \leqslant N-2 \\ \frac{1+\sqrt{8\theta_{k,N}^2+1}}{2} & \text{if } k = N-1, \end{cases}$$

where $\theta_{-1,N} = 0$, and roughly $\theta_{k,N} \approx \frac{k}{2}$.

Optimized gradient method (OGM)

Example I: the optimal method for $\frac{f(y_N)-f_\star}{\|y_0-y_\star\|^2}$ is called the "optimized gradient method" (OGM, Kim & Fessler (2016)):

$$
\begin{aligned}
y_k &= \frac{1}{\theta_{k,N}} z_k + \left(1 - \frac{1}{\theta_{k,N}}\right)\left(y_{k-1} - \tfrac{1}{L} f'(y_{k-1})\right)\\
z_{k+1} &= z_k - \frac{2\theta_{i,N}}{L} f'(y_k)
\end{aligned}
$$

which rely on some exotic sequence

$$
\theta_{k+1,N} = \begin{cases} \frac{1+\sqrt{4\theta_{k,N}^2+1}}{2} & \text{if } k \leqslant N-2 \\ \frac{1+\sqrt{8\theta_{k,N}^2+1}}{2} & \text{if } k = N-1, \end{cases}
$$

where $\theta_{-1,N} = 0$, and roughly $\theta_{k,N} \approx \frac{k}{2}$.

The (tight) worst-case guarantee is

$$
\frac{f(y_N) - f_\star}{L\|y_0 - y_\star\|^2} \leqslant \frac{1}{2\theta_{N,N}^2} \approx \frac{2}{N^2},
$$

which matches exactly the corresponding lower complexity bound (Drori, 2017).

Information-Theoretic Exact Method (ITEM)

Example II: the optimal method for $\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2}$ is called the "Information-Theoretic Exact Method" (ITEM, Drori & T. (2021)). Also simple recurrence

Information-Theoretic Exact Method (ITEM)

Example II: the optimal method for $\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2}$ is called the "Information-Theoretic Exact Method" (ITEM, Drori & T. (2021)). Also simple recurrence

$$y_k = (1-\beta_k)z_k + \beta_k\left(y_{k-1} - \frac{1}{L}f'(y_{k-1})\right)$$

$$z_{k+1} = (1 - \tfrac{\mu}{L}\delta_k)z_k + \tfrac{\mu}{L}\delta_k\left(y_k - \frac{1}{\mu}f'(y_k)\right),$$

Information-Theoretic Exact Method (ITEM)

Example II: the optimal method for $\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2}$ is called the "Information-Theoretic Exact Method" (ITEM, Drori & T. (2021)). Also simple recurrence

$$y_k = (1 - \beta_k) z_k + \beta_k \left(y_{k-1} - \frac{1}{L} f'(y_{k-1}) \right)$$
$$z_{k+1} = (1 - \tfrac{\mu}{L}\delta_k) z_k + \tfrac{\mu}{L}\delta_k \left(y_k - \frac{1}{\mu} f'(y_k) \right),$$

where the sequences $\{\beta_k\}$ and $\{\delta_k\}$ depends on some external sequence

$$A_{k+1} = \frac{(1 + \frac{\mu}{L}) A_k + 2\left(1 + \sqrt{(1 + A_k)(1 + \frac{\mu}{L} A_k)}\right)}{(1 - \frac{\mu}{L})^2}, \quad k \geq 0,$$

with $A_0 = 0$.

Information-Theoretic Exact Method (ITEM)

Example II: the optimal method for $\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2}$ is called the "Information-Theoretic Exact Method" (ITEM, Drori & T. (2021)). Also simple recurrence

$$
\begin{aligned}
y_k &= (1-\beta_k) z_k + \beta_k \left(y_{k-1} - \frac{1}{L} f'(y_{k-1}) \right) \\
z_{k+1} &= (1 - \tfrac{\mu}{L}\delta_k) z_k + \tfrac{\mu}{L}\delta_k \left(y_k - \frac{1}{\mu} f'(y_k) \right),
\end{aligned}
$$

where the sequences $\{\beta_k\}$ and $\{\delta_k\}$ depends on some external sequence

$$
A_{k+1} = \frac{(1+\frac{\mu}{L})A_k + 2\left(1 + \sqrt{(1+A_k)(1+\frac{\mu}{L}A_k)}\right)}{(1-\frac{\mu}{L})^2}, \quad k \geq 0,
$$

with $A_0 = 0$. The (tight) worst-case guarantee is

$$
\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2} \leqslant \frac{1}{1 + \frac{\mu}{L} A_N} = O\left(\left(1 - \sqrt{\frac{\mu}{L}}\right)^{2N} \right),
$$

which matches exactly the corresponding lower complexity bound (Drori & T., 2021).

Notes on those optimal method: OGM & ITEM

How were they found?

Notes on those optimal method: OGM & ITEM

How were they found?

- ◇ by solving the minimax analytically.

Notes on those optimal method: OGM & ITEM

How were they found?

- by solving the minimax analytically.
- (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,

Notes on those optimal method: OGM & ITEM

How were they found?

- ◇ by solving the minimax analytically.
- ◇ (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,
- ◇ (Step 2) lower bounds constructed from interpolation conditions.

Notes on those optimal method: OGM & ITEM

How were they found?

- ⋄ by solving the minimax analytically.
- ⋄ (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,
- ⋄ (Step 2) lower bounds constructed from interpolation conditions.

Relation to Nesterov's method?

Notes on those optimal method: OGM & ITEM

How were they found?

- ◇ by solving the minimax analytically.
- ◇ (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,
- ◇ (Step 2) lower bounds constructed from interpolation conditions.

Relation to Nesterov's method?

- ◇ OGM & ITEM heavily rely on "interpolation inequalities".

Notes on those optimal method: OGM & ITEM

How were they found?

- ◇ by solving the minimax analytically.
- ◇ (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,
- ◇ (Step 2) lower bounds constructed from interpolation conditions.

Relation to Nesterov's method?

- ◇ OGM & ITEM heavily rely on "interpolation inequalities".
- ◇ Those inequalities have caveats, and the methods do not generalize well (e.g., to constraints/nonsmooth term).

Notes on those optimal method: OGM & ITEM

How were they found?

- ◇ by solving the minimax analytically.
- ◇ (Step 1) inspiration from numerical solutions to SDPs & convex relaxations of the minimax problem,
- ◇ (Step 2) lower bounds constructed from interpolation conditions.

Relation to Nesterov's method?

- ◇ OGM & ITEM heavily rely on "interpolation inequalities".
- ◇ Those inequalities have caveats, and the methods do not generalize well (e.g., to constraints/nonsmooth term).
- ◇ Nesterov's method: can be obtained as an optimized gradient method whose proof relies only on more convenient inequalities.

Numerical example

Example III: using PEPs for optimizing a method for $\frac{f(w_N)-f_\star}{f(w_\mathbf{0})-f_\star}$, numerically?

Numerical example

Example III: using PEPs for optimizing a method for $\frac{f(w_N)-f_\star}{f(w_\mathbf{0})-f_\star}$, numerically?

Solving such minimax numerically is NP-hard in general—equivalent to minimization under bilinear matrix inequality (BMI). We work with tractable relaxations.

Numerical example

Example III: using PEPs for optimizing a method for $\frac{f(w_N)-f_\star}{f(w_{\mathbf{0}})-f_\star}$, numerically?

Solving such minimax numerically is NP-hard in general—equivalent to minimization under bilinear matrix inequality (BMI). We work with tractable relaxations.

Let us pick $L = 1$, $\mu = .1$. The best method we could reach (through numerical optimization) for $N = 5$ satisfies

$$\frac{f(w_{\mathbf{5}})-f_\star}{f(w_{\mathbf{0}})-f_\star} \leq 0.0365,$$

Numerical example

Example III: using PEPs for optimizing a method for $\frac{f(w_N)-f_\star}{f(w_0)-f_\star}$, numerically?

Solving such minimax numerically is NP-hard in general—equivalent to minimization under bilinear matrix inequality (BMI). We work with tractable relaxations.

Let us pick $L = 1$, $\mu = .1$. The best method we could reach (through numerical optimization) for $N = 5$ satisfies

$$\frac{f(w_5)-f_\star}{f(w_0)-f_\star} \leq 0.0365,$$

with step sizes

$$[h^\star_{i,j}] = \begin{bmatrix} 1.9060 & & & & \\ 0.3879 & 2.1439 & & & \\ 0.1585 & 0.4673 & 2.1227 & & \\ 0.0660 & 0.1945 & 0.4673 & 2.1439 & \\ 0.0224 & 0.0660 & 0.1585 & 0.3879 & 1.9060 \end{bmatrix}.$$

Numerical example

This can be done for a few values of L, μ and N. For example (still $L = 1$, $\mu = .1$)

Numerical example

This can be done for a few values of L, μ and N. For example (still $L = 1$, $\mu = .1$)

- For a single iteration, $N = 1$, we obtain a guarantee $\frac{f(w_{\mathbf{1}})-f_\star}{f(w_{\mathbf{0}})-f_\star} \leq 0.6694$ with the corresponding step size
$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix},$$
which matches the known optimal step size $2/(L+\mu)$.

Numerical example

This can be done for a few values of L, μ and N. For example (still $L = 1$, $\mu = .1$)

- For a single iteration, $N = 1$, we obtain a guarantee $\frac{f(w_1)-f_\star}{f(w_0)-f_\star} \leq 0.6694$ with the corresponding step size

$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix},$$

  which matches the known optimal step size $2/(L+\mu)$.

- For $N = 2$, we obtain $\frac{f(w_2)-f_\star}{f(w_0)-f_\star} \leq 0.3554$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 2.0095 & \\ 0.4229 & 2.0095 \end{bmatrix}.$$

Numerical example

This can be done for a few values of L, μ and N. For example (still $L = 1$, $\mu = .1$)

- For a single iteration, $N = 1$, we obtain a guarantee $\frac{f(w_1)-f_\star}{f(w_0)-f_\star} \leq 0.6694$ with the corresponding step size
$$[h^\star_{i,j}] = \begin{bmatrix}1.8182\end{bmatrix},$$
which matches the known optimal step size $2/(L+\mu)$.
- For $N = 2$, we obtain $\frac{f(w_2)-f_\star}{f(w_0)-f_\star} \leq 0.3554$ with
$$[h^\star_{i,j}] = \begin{bmatrix}2.0095 & \\ 0.4229 & 2.0095\end{bmatrix}.$$
- For $N = 3$, we obtain $\frac{f(w_3)-f_\star}{f(w_0)-f_\star} \leq 0.1698$ with
$$[h^\star_{i,j}] = \begin{bmatrix}1.9470 & & \\ 0.4599 & 2.2406 & \\ 0.1705 & 0.4599 & 1.9470\end{bmatrix}.$$

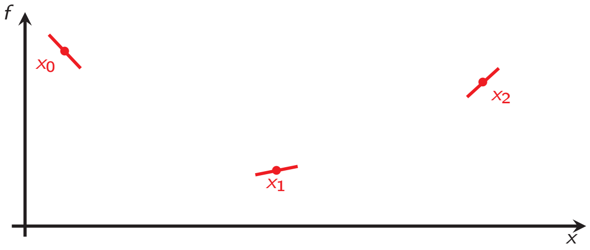
Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!

Reminder: smooth strongly convex interpolation/extension

Consider a set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, subgradients g_i and function values f_i.



? Possible to find a $f \in \mathcal{F}_{\mu,L}$ s.t.

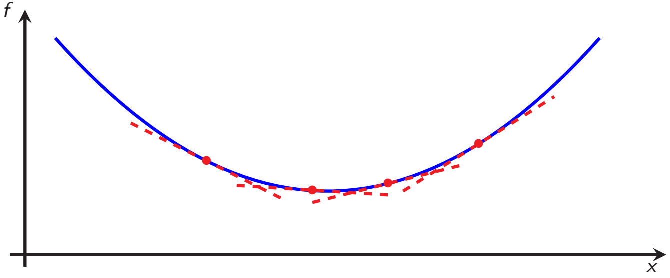
$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?

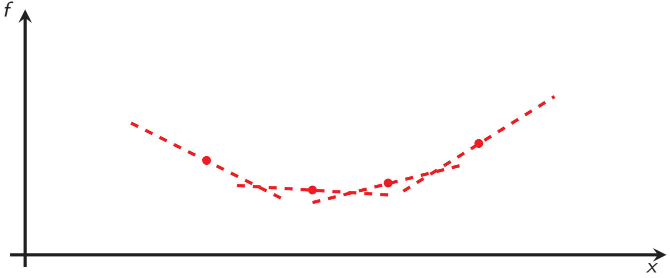
Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i\in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Special case: convex interpolation problem

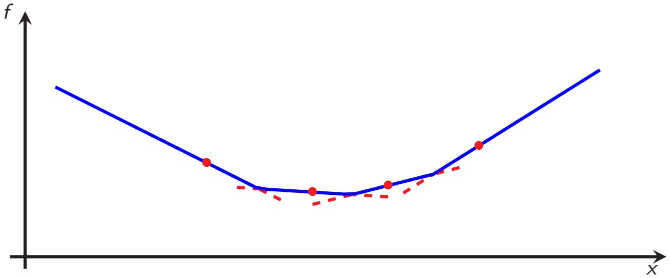
Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Conditions $f_i \geq f_j + \langle g_j, x_i - x_j \rangle$ is nec.

Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Conditions $f_i \geq f_j + \langle g_j, x_i - x_j \rangle$ is nec. and suff.

Explicit construction:

$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\},$$

Not unique.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.
- ◇ We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- ◇ Example: (ccp) convex minimization, worst-case problems can be assumed to have the form

$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- Similar constructions for smooth (possibly) strongly convex functions.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.
- ◇ We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- ◇ Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- ◇ Similar constructions for smooth (possibly) strongly convex functions.
- ◇ We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.
- ◇ We can infer shapes for the worst-case functions!
  - – Why? Let's flashback into the interpolation/extension problem!
- ◇ Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- ◇ Similar constructions for smooth (possibly) strongly convex functions.
- ◇ We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - – idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.
- ◇ We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- ◇ Example: (ccp) convex minimization, worst-case problems can be assumed to have the form

$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$

- ◇ Similar constructions for smooth (possibly) strongly convex functions.
- ◇ We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;
  - those constraints fit into a SDP;

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form

$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$

- Similar constructions for smooth (possibly) strongly convex functions.
- We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;
  - those constraints fit into a SDP;
  - such functions are sometimes referred to as being "zero-chain".

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),

Numerical example

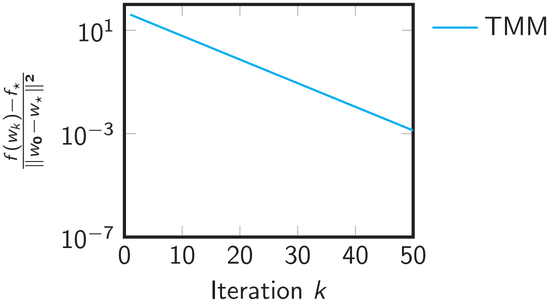
Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

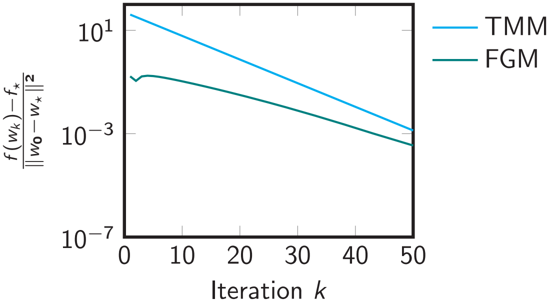
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L = 1$ and $\mu = .01$. We compare

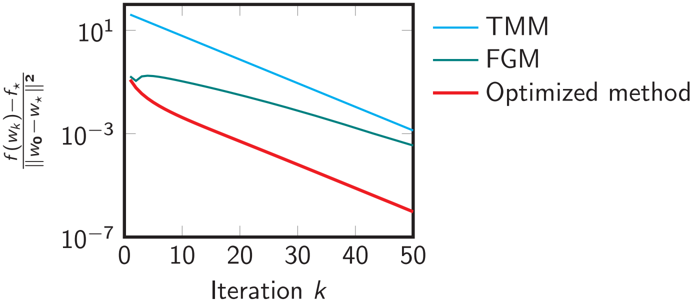
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

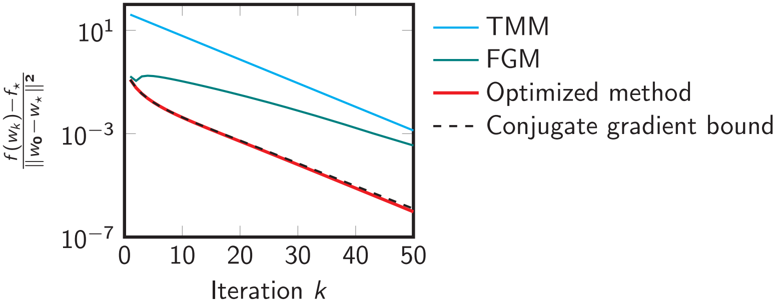
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

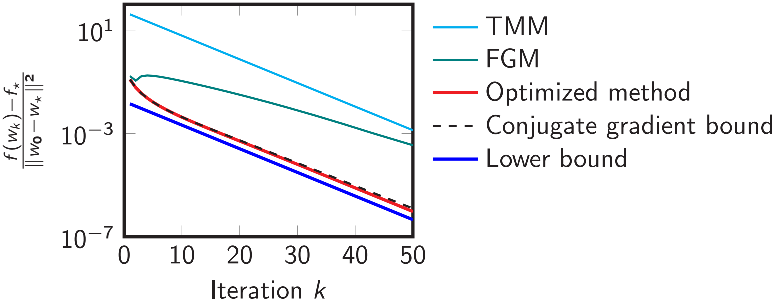
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{\|w_0-w_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L = 1$ and $\mu = .01$. We compare

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_\mathbf{0})-f_\star}$ with $L=1$ and $\mu=.01$. We compare

- $\diamond$ worst-case performance of optimized method (numerically generated),

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),

Numerical example

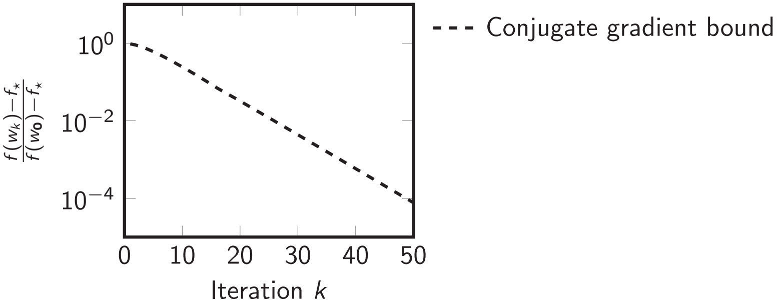
Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L=1$ and $\mu=.01$. We compare

- ◇ worst-case performance of optimized method (numerically generated),
- ◇ conjugate-gradient based method (numerically generated),
- ◇ lower complexity bound (numerically generated).

Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L = 1$ and $\mu = .01$. We compare

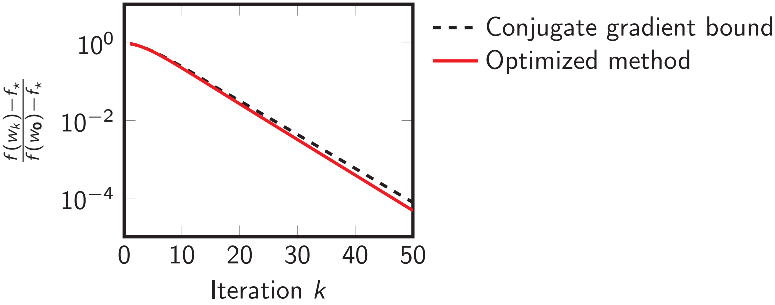
- ◇ worst-case performance of optimized method (numerically generated),
- ◇ conjugate-gradient based method (numerically generated),
- ◇ lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L = 1$ and $\mu = .01$. We compare

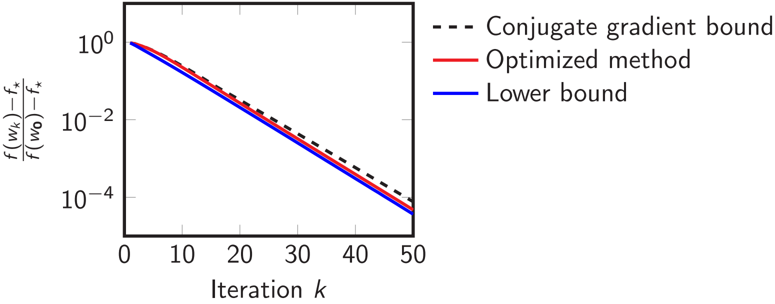
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- lower complexity bound (numerically generated).



Numerical example

Worst-case performance $\frac{f(w_k)-f_\star}{f(w_0)-f_\star}$ with $L = 1$ and $\mu = .01$. We compare

- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- lower complexity bound (numerically generated).



A few observations/limitations

Were we lucky? Some pieces might be missing!

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- ◇ $\frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2}$ with $\mu = 0$: optimized gradient method (OGM, Kim & Fessler 2016),

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|w_N-w_\star\|^2}{\|w_0-w_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(w_N)-f_\star}{\|w_0-w_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|w_N-w_\star\|^2}{\|w_0-w_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- $\frac{\|f'(w_N)\|^2}{f(w_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

Performance estimation problems

Designing methods using PEPs

Conclusions

Concluding remarks

Performance estimation's philosophy

Concluding remarks

Performance estimation's philosophy

- ◇ numerically allows obtaining tight bounds (rigorous baselines),

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- fast prototyping:

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- fast prototyping:

  before trying to prove your new FO method works; give PEP a try!

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- fast prototyping:

  before trying to prove your new FO method works; give PEP a try!
- step forward to "reproducible theory" (useful for reviewing, too ☺)

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- fast prototyping:

  before trying to prove your new FO method works; give PEP a try!
- step forward to "reproducible theory" (useful for reviewing, too ☺)
- overall: principled approach (definition of worst-case).

Concluding remarks

Performance estimation's philosophy

- ◇ numerically allows obtaining tight bounds (rigorous baselines),
- ◇ results can only be improved by changing algorithm and/or assumptions,
- ◇ helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- ◇ fast prototyping:

  before trying to prove your new FO method works; give PEP a try!
- ◇ step forward to "reproducible theory" (useful for reviewing, too ☺)
- ◇ overall: principled approach (definition of worst-case).

Difficulties:

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- results can only be improved by changing algorithm and/or assumptions,
- helps designing analytical proofs (reduces to linear combinations of inequalities),
  proofs can be engineered using numerics & symbolic computations!
- fast prototyping:
  before trying to prove your new FO method works; give PEP a try!
- step forward to "reproducible theory" (useful for reviewing, too ☺)
- overall: principled approach (definition of worst-case).

Difficulties:

- suffers from standard caveats of worst-case analyses,
  key is to find good assumptions/parametrization

Concluding remarks

Performance estimation's philosophy

- ◇ numerically allows obtaining tight bounds (rigorous baselines),
- ◇ results can only be improved by changing algorithm and/or assumptions,
- ◇ helps designing analytical proofs (reduces to linear combinations of inequalities),

  proofs can be engineered using numerics & symbolic computations!
- ◇ fast prototyping:

  before trying to prove your new FO method works; give PEP a try!
- ◇ step forward to "reproducible theory" (useful for reviewing, too ☺)
- ◇ overall: principled approach (definition of worst-case).

Difficulties:

- ◇ suffers from standard caveats of worst-case analyses,

  key is to find good assumptions/parametrization
- ◇ closed-form solutions might be involved.

Take-home messages

Worst-cases are solutions to optimization problems.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

When it does: principled approach to worst-case analyses.

Thanks! Questions?

www.di.ens.fr/~ataylor/

AdrienTaylor/Performance-Estimation-Toolbox on Github