Convex Optimization for Big Data

Volkan Cevher,[⋆] **Mário Figueiredo**,[†] **Mark Schmidt**,[‡] *and* **Quoc Tran-Dinh**[⋆]

[⋆]Laboratory for Information and Inference Systems
École Polytechnique Fédérale de Lausanne (EPFL)

[†]Instituto de Telecomunicações, Instituto Superior Técnico, University of Lisbon

[‡]Laboratory for Computational Intelligence, University of British Columbia

The 40th IEEE International Conference on Acoustics, Speech and Signal Processing

ICASSP 2015 (19 - 24 April 2015) Brisbane, Australia

lions@epfl

License Information for “Convex Optimization for Big Data” Slides

Acknowledgements

Quite a few people contributed to these slides

- **@LIONS**: Bubacarr Bah, Luca Baldassarre, Marwa El-Halabi, Baran Gozcu, Radu-Christian Ionescu, Anastasios Kyrillidis, Yen-Huan Li, Jonathan Scarlett, and Alp Yurtsever
- @Colorado: Stephen R. Becker
- @Caltech: Michael B. McCoy

... and we are still working!!! You can find the updated version at

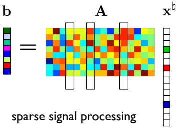
`http://lions.epfl.ch/teaching/tutorials`

Related materials/tutorials:

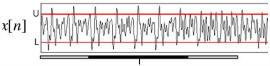
- V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data," *IEEE Signal Processing Magazine*, 2014.
  Available at `lions.epfl.ch/publications`
- V. Cevher and M. Figueiredo, "Convex and Non-convex Approaches for Low-dimensional Models," tutorial at *ICASSP 2012*.
  Available at `lions.epfl.ch/teaching/tutorials`
- M. Figueiredo and S. Wright, "Sparse Optimization and Applications to Information Processing," tutorial at ICCOPT 2013.
  Available at `www.lx.it.pt/~mtf/#talks`

The key role of convex optimization in (Big) data sciences

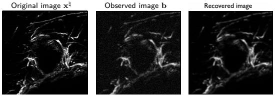
From SP and geophysics to medical and hyperspectral imaging



sparse signal processing



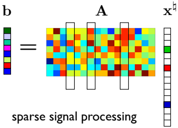
audio-signal processing



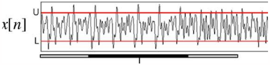
medical imaging

The key role of convex optimization in (Big) data sciences

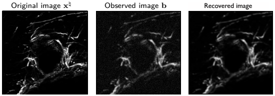
From SP and geophysics to medical and hyperspectral imaging



sparse signal processing



audio-signal processing



medical imaging

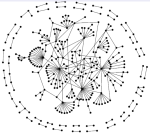
From machine learning and NLP to statistics and bioinformatics

NETFLIX

Recommendation systems

facebook

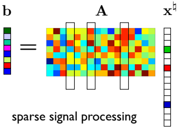
Topic models



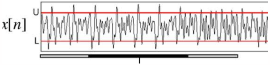
Graphical model selection

The key role of convex optimization in (Big) data sciences

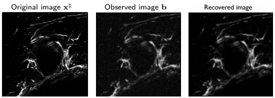
From SP and geophysics to medical and hyperspectral imaging



sparse signal processing



audio-signal processing

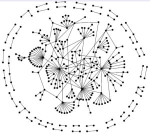


medical imaging

From machine learning and NLP to statistics and bioinformatics

Recommendation systems

Topic models



Graphical model selection

From control and power systems to network science

Predictive control, system identification, controller design,...
Google page rank, social networks, transportation networks, power grids, utilities,...

Challenges for convex optimization

- High ambient dimension p and "big" data n
- Non-smooth objectives and constraint sets
- Increasingly elaborate observation models

Challenges for convex optimization

- High ambient dimension p and "big" data n
- Non-smooth objectives and constraint sets
- Increasingly elaborate observation models

This tutorial:

exploiting structures in optimization and stochastic approximation

Warm up: *Convexity*

Definition (Convex function)

$f: \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\} = \bar{\mathbb{R}}$ is said to be convex if, for any $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom}(f)$ and $\alpha \in [0,1]$,

$$f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \le \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2).$$

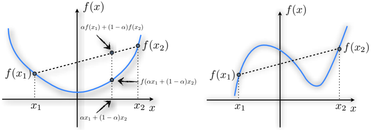


Figure: (**Left**) Convex (**Right**) Non-convex

Warm up: *Convexity*

Definition (Convex function)

$f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\} = \bar{\mathbb{R}}$ is said to be convex if, for any $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom}(f)$ and $\alpha \in [0,1]$,

$$f(\alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \le \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2).$$

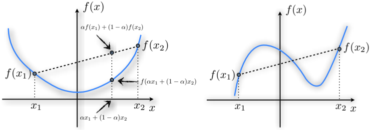


Figure: (**Left**) Convex (**Right**) Non-convex

Definition (Convex set)

$\mathcal{Q} \subseteq \mathbb{R}^p$ is convex if,
$\mathbf{x}_1,\ \mathbf{x}_2 \in \mathcal{Q} \Rightarrow \alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in \mathcal{Q}, \forall \alpha \in [0,1]$

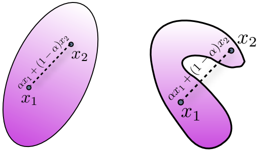


Figure: (**Left**) Convex (**Right**) Non-convex

Warm up: *Convexity*

Definition (Convex function)

$f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\} = \bar{\mathbb{R}}$ is said to be convex if, for any $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom}(f)$ and $\alpha \in [0, 1]$,

$$f(\alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2).$$

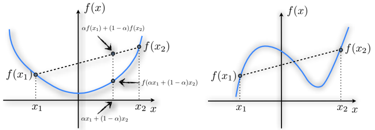


Figure: (**Left**) Convex (**Right**) Non-convex

Definition (Convex set)

$\mathcal{Q} \subseteq \mathbb{R}^p$ is convex if,
$\mathbf{x}_1,\ \mathbf{x}_2 \in \mathcal{Q} \Rightarrow \alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in \mathcal{Q}, \forall \alpha \in [0, 1]$

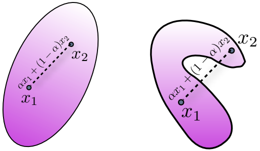


Figure: (**Left**) Convex (**Right**) Non-convex

Role/importance of convexity:

- Useful in optimization
  - local minima are global
- Convex programs
  - relaxations of non-convex problems with rigorous guarantees
- Tractability
  - often (not always) polynomial time

cf. Lecture 2 @
`http://lions.epfl.ch/mathematics_of_data`

Warm up: ***Norms as convex functions***

Definition (Norm)

A function $f : \mathbb{R}^p \to \mathbb{R}$ is a norm if, for all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, and scalar $\lambda \in \mathbb{R}$,

(a)	$f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^p$	(*nonnegativity*)
(b)	$f(\mathbf{x}) = 0$ if and only if $\mathbf{x} = \mathbf{0}$	(*definitiveness*)
(c)	$f(\lambda\mathbf{x}) = \lvert\lambda\rvert f(\mathbf{x})$	(*homogeneity*)
(d)	$f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$	(*triangle inequality*)

Warm up: *Norms as convex functions*

Definition (Norm)

A function $f : \mathbb{R}^p \to \mathbb{R}$ is a norm if, for all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, and scalar $\lambda \in \mathbb{R}$,

(a)	$f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^p$	(*nonnegativity*)
(b)	$f(\mathbf{x}) = 0$ if and only if $\mathbf{x} = \mathbf{0}$	(*definitiveness*)
(c)	$f(\lambda \mathbf{x}) = \lvert\lambda\rvert f(\mathbf{x})$	(*homogeneity*)
(d)	$f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$	(*triangle inequality*)

The q-norms for vectors:

1. ℓ_q-norm $(q \geq 1)$: $\|\mathbf{x}\|_q := \left[\sum_{i=1}^p |x_i|^q\right]^{1/q}$, and $\|\mathbf{x}\|_\infty = \max_i |x_i|$.
2. Inner product: $\mathbf{x}^T\mathbf{y} \equiv \langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^p x_i y_i, \quad \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|_2^2$.

Warm up: *Norms as convex functions*

Definition (Norm)

A function $f : \mathbb{R}^p \to \mathbb{R}$ is a norm if, for all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, and scalar $\lambda \in \mathbb{R}$,

(a)	$f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^p$	(*nonnegativity*)
(b)	$f(\mathbf{x}) = 0$ if and only if $\mathbf{x} = \mathbf{0}$	(*definitiveness*)
(c)	$f(\lambda\mathbf{x}) = \lvert\lambda\rvert f(\mathbf{x})$	(*homogeneity*)
(d)	$f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$	(*triangle inequality*)

The q-norms for vectors:

1. ℓ_q-norm $(q \geq 1)$: $\|\mathbf{x}\|_q := \left[\sum_{i=1}^p |x_i|^q\right]^{1/q}$, and $\|\mathbf{x}\|_\infty = \max_i |x_i|$.
2. Inner product: $\mathbf{x}^T\mathbf{y} \equiv \langle \mathbf{x}, \mathbf{y} \rangle := \sum_{i=1}^p x_i y_i, \quad \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|_2^2$.

The Schatten q-norms for matrices:

1. Schatten q-norms: $\|\mathbf{A}\|_{S_q} := \left(\sum_{i=1}^p \sigma_i(\mathbf{A})^q\right)^{1/q}$, and $\|\mathbf{A}\|_{S_\infty} = \sigma_1(\mathbf{A})$ where $\sigma_i(\mathbf{A})$ is the i^{th} largest singular value of $\mathbf{A}$.
2. Inner product: $\langle \mathbf{A}, \mathbf{B} \rangle := \mathrm{trace}(\mathbf{A}^T\mathbf{B}) = \langle \mathrm{vec}(\mathbf{A}), \mathrm{vec}(\mathbf{B}) \rangle$, $\langle \mathbf{A}, \mathbf{A} \rangle = \|\mathbf{A}\|_F^2$.

cf. Lecture 1 @ `http://lions.epfl.ch/mathematics_of_data`

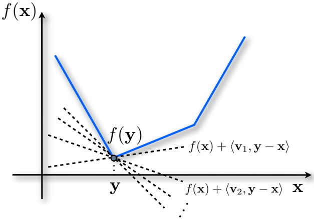
Warm up: *(Non-)Smoothness in convex functions*

Definition (Subdifferential)

$\mathbf{v} \in \mathbb{R}^p$ is a subgradient of convex function $f : \mathcal{Q} \to \bar{\mathbb{R}}$, at $\mathbf{x} \in \mathcal{Q}$, if, $\forall \mathbf{y} \in \mathcal{Q}$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{v},\ \mathbf{y} - \mathbf{x} \rangle$$

The subdifferential of f at $\mathbf{x}$, denoted $\partial f(\mathbf{x})$ is the set of all subgradients.



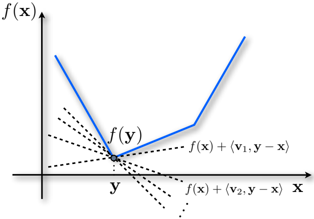
Warm up: *(Non-)Smoothness in convex functions*

Definition (Subdifferential)

$\mathbf{v} \in \mathbb{R}^p$ is a subgradient of convex function $f : \mathcal{Q} \to \bar{\mathbb{R}}$, at $\mathbf{x} \in \mathcal{Q}$, if, $\forall \mathbf{y} \in \mathcal{Q}$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{v},\ \mathbf{y} - \mathbf{x} \rangle$$

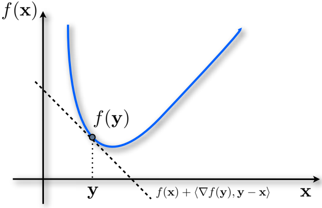
The subdifferential of f at $\mathbf{x}$, denoted $\partial f(\mathbf{x})$ is the set of all subgradients.



Proposition (Gradient)

If $f : \mathcal{Q} \to \bar{\mathbb{R}}$ is differentiable and convex, then,

$$\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}\,.$$



Warm up: *(Non-)Smoothness in convex functions*

Definition (Subdifferential)

$\mathbf{v} \in \mathbb{R}^p$ is a subgradient of convex function $f : \mathcal{Q} \to \bar{\mathbb{R}}$, at $\mathbf{x} \in \mathcal{Q}$, if, $\forall \mathbf{y} \in \mathcal{Q}$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{v}, \ \mathbf{y} - \mathbf{x} \rangle$$

The subdifferential of f at $\mathbf{x}$, denoted $\partial f(\mathbf{x})$ is the set of all subgradients.

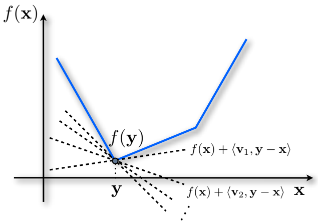
Proposition (Gradient)

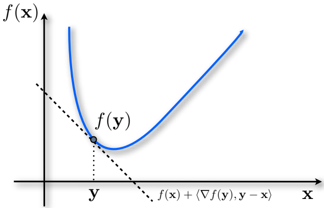
If $f : \mathcal{Q} \to \bar{\mathbb{R}}$ is differentiable and convex, then,

$$\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}.$$

Role of non-smoothness:

- Useful in modeling
  - sparsity, low-rank, TV...
- Convex optimization
  - significantly inefficient vs. smooth





cf. Lecture 1 & 2 @ `http://lions.epfl.ch/mathematics_of_data`

Towards Big Data: A simple *regression* model

$$\boxed{\mathbf{b}_i = \mathbf{x}^\natural(\mathbf{a}_i) + \mathbf{w}_i}$$

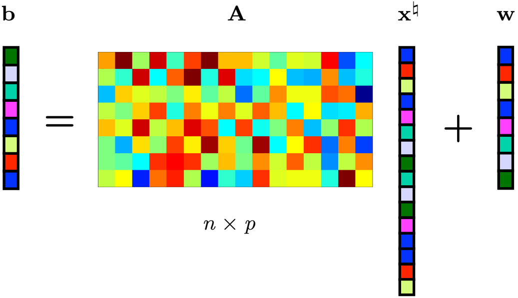
$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

Towards Big Data: A simple *regression* model

$$\boxed{\mathbf{b}_i = \mathbf{x}^\natural(\mathbf{a}_i) + \mathbf{w}_i}$$

$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

Linear model:



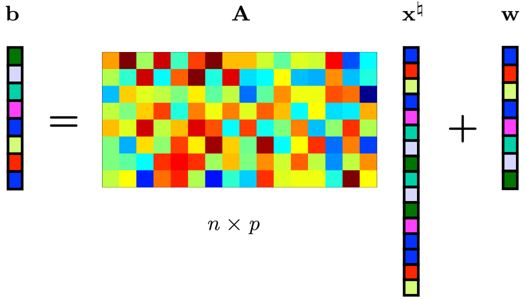
$$\boxed{\mathbf{b}_i = \mathbf{x}^\natural(\mathbf{a}_i) + \mathbf{w}_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{w}_i}$$

Towards Big Data: A simple *regression* model

$$\boxed{\mathbf{b}_i = \mathbf{x}^\natural(\mathbf{a}_i) + \mathbf{w}_i}$$

$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

Linear model:



$$\boxed{\mathbf{b}_i = \mathbf{x}^\natural(\mathbf{a}_i) + \mathbf{w}_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{w}_i}$$

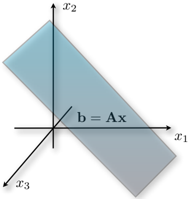
Applications: **Compressive sensing, machine learning, theoretical computer science...**

A simple *regression* model and many *practical* questions

$$\boxed{\mathbf{b}_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{w}_i}$$

$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

- Estimation: find $\mathbf{x}^\star$ to minimize $\|\mathbf{x}^\star - \mathbf{x}^\natural\|$
- Prediction: find $\mathbf{x}^\star$ to minimize $\mathbb{E}_{\mathbf{a},\mathbf{w}}\,\mathcal{L}\left(\mathbf{x}^\star(\mathbf{a}), \mathbf{x}^\natural(\mathbf{a}) + \mathbf{w}\right)$
- Decision: choose $\mathbf{a}_i$ for estimation or prediction

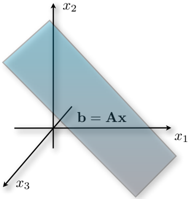


A simple *regression* model and many *practical* questions

$$\boxed{\mathbf{b}_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{w}_i}$$

$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

- Estimation: find $\mathbf{x}^\star$ to minimize $\|\mathbf{x}^\star - \mathbf{x}^\natural\|$
- Prediction: find $\mathbf{x}^\star$ to minimize $\mathbb{E}_{\mathbf{a},\mathbf{w}}\, \mathcal{L}\left(\mathbf{x}^\star(\mathbf{a}), \mathbf{x}^\natural(\mathbf{a}) + \mathbf{w}\right)$
- Decision: choose $\mathbf{a}_i$ for estimation or prediction



A difficult estimation challenge when $n < p$:

Nullspace (null) of $\mathbf{A}$: $\quad \mathbf{x}^\natural + \boldsymbol{\delta} \rightarrow \mathbf{b}, \quad \forall \boldsymbol{\delta} \in \text{null}(\mathbf{A})$

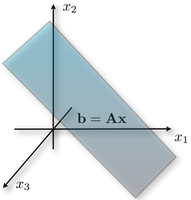
- **Needle in a haystack:** ***We need additional information on $\mathbf{x}^\natural$!***

A simple *regression* model and many *practical* questions

$$\boxed{\mathbf{b}_i = \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle + \mathbf{w}_i}$$

$\mathbf{x}^\natural$: unknown function / hypothesis
$\mathbf{a}_i$: input
$\mathbf{b}_i$: response / output
$\mathbf{w}_i$: perturbations / noise

- Estimation: find $\mathbf{x}^\star$ to minimize $\|\mathbf{x}^\star - \mathbf{x}^\natural\|$
- Prediction: find $\mathbf{x}^\star$ to minimize $\mathbb{E}_{\mathbf{a},\mathbf{w}}\,\mathcal{L}\left(\mathbf{x}^\star(\mathbf{a}), \mathbf{x}^\natural(\mathbf{a}) + \mathbf{w}\right)$
- Decision: choose $\mathbf{a}_i$ for estimation or prediction



A difficult estimation challenge when $n < p$***:***

Nullspace (null) of $\mathbf{A}$: $\quad \mathbf{x}^\natural + \boldsymbol{\delta} \rightarrow \mathbf{b}, \quad \forall \boldsymbol{\delta} \in \text{null}(\mathbf{A})$

- **Needle in a haystack:** ***We need additional information on*** $\mathbf{x}^\natural$***!***

A difficult computational challenge when n ***and*** p ***are large:***

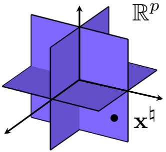
- ***We need scalable algorithms!***

Swiss army knife of signal models

Definition (s-sparse vector)

A vector $\mathbf{x} \in \mathbb{R}^p$ is s-sparse, i.e., $\mathbf{x} \in \Sigma_s$, if it has at most s non-zero entries.

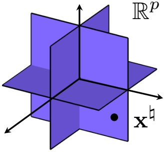
$$\left\|\mathbf{x}^\natural\right\|_0 := |\{i : x_i^\natural \neq 0\}| = s$$



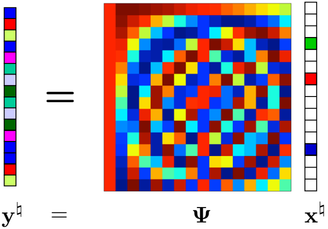
Swiss army knife of signal models

Definition (s-sparse vector)

A vector $\mathbf{x} \in \mathbb{R}^p$ is s-sparse, i.e., $\mathbf{x} \in \Sigma_s$, if it has at most s non-zero entries.



$$\left\|\mathbf{x}^\natural\right\|_0 := |\{i : x_i^\natural \neq 0\}| = s$$



Sparse representations:

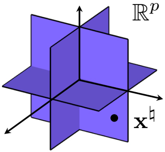
$\mathbf{y}^\natural$ has *sparse* transform coefficients $\mathbf{x}^\natural$

- Basis representations $\Psi \in \mathbb{R}^{p \times p}$
  - *Wavelets*, DCT, ...
- Frame representations $\Psi \in \mathbb{R}^{m \times p}$, $m > p$
  - Gabor, curvelets, shearlets, ...
- Other *dictionary* representations...

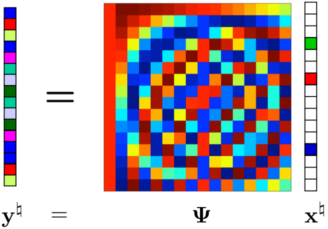
Swiss army knife of signal models

Definition (s-sparse vector)

A vector $\mathbf{x} \in \mathbb{R}^p$ is s-sparse, i.e., $\mathbf{x} \in \Sigma_s$, if it has at most s non-zero entries.



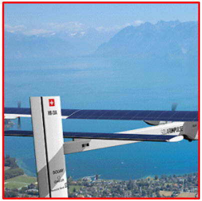
$$\left\|\mathbf{x}^\natural\right\|_0 := |\{i : x_i^\natural \neq 0\}| = s$$

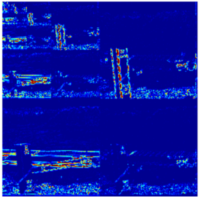


Sparse representations:

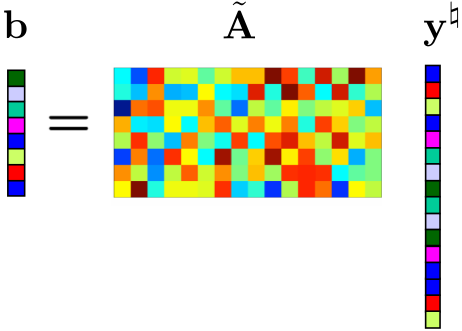
$\mathbf{y}^\natural$ has *sparse* transform coefficients $\mathbf{x}^\natural$

- Basis representations $\Psi \in \mathbb{R}^{p \times p}$
  - *Wavelets*, DCT, ...
- Frame representations $\Psi \in \mathbb{R}^{m \times p}$, $m > p$
  - Gabor, curvelets, shearlets, ...
- Other *dictionary* representations...



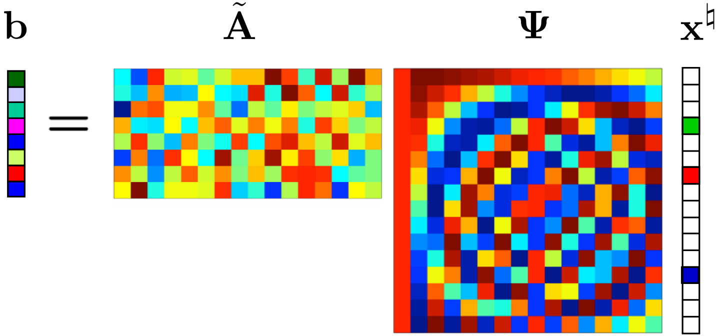


Sparse representations strike back!



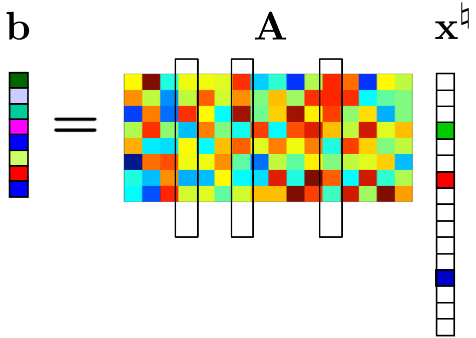
- $\mathbf{b} \in \mathbb{R}^n$, $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times p}$, and $n < p$

Sparse representations strike back!



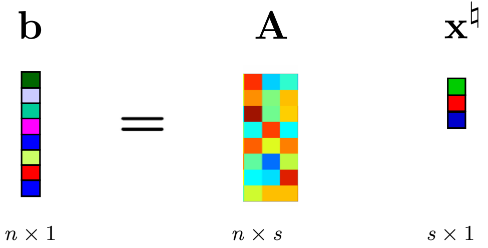
- $\mathbf{b} \in \mathbb{R}^n$, $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times p}$, and $n < p$
- $\boldsymbol{\Psi} \in \mathbb{R}^{p \times p}$, $\mathbf{x}^\natural \in \mathbb{R}^p$, and $\|\mathbf{x}^\natural\|_0 \leq s < n$

Sparse representations strike back!



- $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times p}$, and $\mathbf{x}^\natural \in \mathbb{R}^p$, and $\|\mathbf{x}^\natural\|_0 \leq s < n < p$

Sparse representations strike back!



- $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times p}$, and $\mathbf{x}^\natural \in \mathbb{R}^p$, and $\|\mathbf{x}^\natural\|_0 \leq s < n < p$

Impact: Support restricted columns of $\mathbf{A}$ leads to an *overcomplete* system.

Enter sparsity

A combinatorial approach for estimating $\mathbf{x}^\natural$ from $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$

We may consider the estimator with the least number of non-zero entries. That is,

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_0 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \kappa \right\} \qquad (\mathcal{P}_0)$$

with some $\kappa \geq 0$. If $\kappa = \|\mathbf{w}\|_2$, then $\mathbf{x}^\natural$ is a feasible solution.

Enter sparsity

A combinatorial approach for estimating $\mathbf{x}^\natural$ from $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$

We may consider the estimator with the least number of non-zero entries. That is,

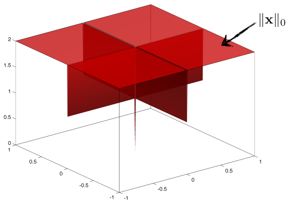
$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_0 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \kappa \right\} \qquad (\mathcal{P}_0)$$

with some $\kappa \geq 0$. If $\kappa = \|\mathbf{w}\|_2$, then $\mathbf{x}^\natural$ is a feasible solution.

$\mathcal{P}_0$ has the following characteristics:

- sample complexity: $\mathcal{O}(s)$
- computational effort: NP-Hard
- stability: No

$\|\mathbf{x}\|_0$ over the unit ℓ_∞-ball



Enter sparsity

A combinatorial approach for estimating $\mathbf{x}^\natural$ from $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$

We may consider the estimator with the least number of non-zero entries. That is,

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_0 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \kappa \right\} \qquad (\mathcal{P}_0)$$

with some $\kappa \geq 0$. If $\kappa = \|\mathbf{w}\|_2$, then $\mathbf{x}^\natural$ is a feasible solution.

$\mathcal{P}_0$ has the following characteristics:

- sample complexity: $\mathcal{O}(s)$
- computational effort: NP-Hard
- stability: No

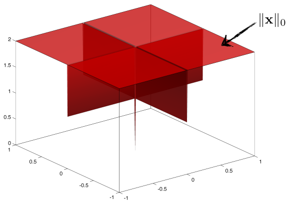
A convex relaxation:

$\|\mathbf{x}\|_0^{**}$ as the biconjugate (Fenchel conjugate of Fenchel conjugate) of $\|\mathbf{x}\|_0$ over the unit ℓ_∞-ball.

Fenchel conjugate:

$$f^*(\mathbf{y}) := \sup_{\mathbf{x}\in\mathrm{dom}(f)} \left\{ \mathbf{x}^T\mathbf{y} - f(\mathbf{x}) \right\}.$$

$\|\mathbf{x}\|_0$ over the unit ℓ_∞-ball



Enter sparsity

A combinatorial approach for estimating $\mathbf{x}^\natural$ from $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$

We may consider the estimator with the least number of non-zero entries. That is,

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_0 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \kappa \right\} \qquad (\mathcal{P}_0)$$

with some $\kappa \geq 0$. If $\kappa = \|\mathbf{w}\|_2$, then $\mathbf{x}^\natural$ is a feasible solution.

$\mathcal{P}_0$ has the following characteristics:

- sample complexity: $\mathcal{O}(s)$
- computational effort: NP-Hard
- stability: No

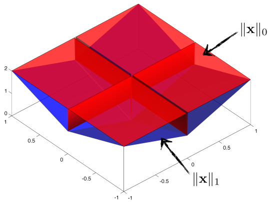
A convex relaxation:

$\|\mathbf{x}\|_0^{**}$ as the biconjugate (Fenchel conjugate of Fenchel conjugate) of $\|\mathbf{x}\|_0$ over the unit ℓ_∞-ball.

Fenchel conjugate:

$$f^*(\mathbf{y}) := \sup_{\mathbf{x}\in\mathrm{dom}(f)} \left\{ \mathbf{x}^T\mathbf{y} - f(\mathbf{x}) \right\}.$$

$\|\mathbf{x}\|_1$ is the convex envelope of $\|\mathbf{x}\|_0$



The role of convexity

A convex candidate solution for $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_1 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{w}\|_2, \|\mathbf{x}\|_\infty \leq 1 \right\}. \qquad \text{(SOCP)}$$

Theorem (A model recovery guarantee [53])

Let $\mathbf{A} \in \mathbb{R}^{n\times p}$ be a matrix of i.i.d. Gaussian random variables with zero mean and variances $1/n$. For any $t > 0$ with probability at least $1 - 6\exp\left(-t^2/26\right)$, we have

$$\left\|\mathbf{x}^\star - \mathbf{x}^\natural\right\|_2 \leq \left[\frac{2\sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s}}{\sqrt{n} - \sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s} - t}\right] \|\mathbf{w}\|_2 := \varepsilon, \quad \textit{when } \|\mathbf{x}^\natural\|_0 \leq s.$$

Observations:

- perfect recovery (i.e., $\varepsilon = 0$) with $n \geq 2s\log(\frac{p}{s}) + \frac{5}{4}s$ whp when $\mathbf{w} = 0$.
- ϵ-accurate solution in $k = \mathcal{O}\left(\sqrt{2p+1}\log(\frac{1}{\epsilon})\right)$ iterations via IPM[1] using matrix-matrix / matrix-vector operations with matrix sizes $n \times 2p$.[2]
- robust to noise.

---

[1] There is a subtle yet important caveat here that I am sweeping under the carpet!

[2] When $\mathbf{w} = 0$, the IPM complexity (# of iterations $\times$ cost per iteration) amounts to $\mathcal{O}(n^2 p^{1.5}\log(\frac{1}{\epsilon}))$.

Complexity of basic computational primitives

Definition (floating-point operation)

A **floating-point operation** (flop) is one addition, subtraction, multiplication, or division of two floating-point numbers.[3]

Table: Complexity examples; vectors are in $\mathbb{R}^p$. Matrices are in $\mathbb{R}^{m\times n}$ or $\mathbb{R}^{n\times p}$ or $\mathbb{R}^{p\times p}$.

Operation	Complexity	Remarks
vector addition	p flops	
vector inner product	$2p-1$ flops	or $\approx 2p$ for p large
matrix-vector product	$n(2p-1)$ flops	or $\approx 2np$ for p large $2m$ if $\mathbf{A}$ is sparse with m nonzeros
matrix-matrix product	$mn(2p-1)$ flops	or $\approx 2mnp$ for p large (naïve method) much less if the matrices are sparse[1,2]
LU decomposition	$\frac{2}{3}p^3 + 2p^2$ flops	or $\approx \frac{2}{3}p^3$ for p large much less if the matrix is sparse[1]
Cholesky decomposition	$\frac{1}{3}p^3 + 2p^2$ flops	or $\approx \frac{1}{3}p^3$ for p large much less if the matrix is sparse[1]
Matrix SVD	$C_1 n^2 p + C_2 p^3$ flops	$C_1 = 4,\ C_2 = 22$ for R-SVD algo.
Matrix determinant	complexity of SVD$+p$ flops	much less for sparse $\mathbf{A}$ using Cholesky
Matrix inverse	$Cp^{\log_2 7}$ flops,	$4 < C < 5$ using Strassen algorithm

[1] Computational complexity depends on the number of nonzeros in the matrices.
[2] For multiplying $p \times p$ matrices, the best computational complexity result is currently $O(p^{2.373})$.

[3] In computing, flops, i.e., the plural form of flop, also stands for FLoating-point Operations Per Second, which measures the rate. We can disambiguate depending on the context.

A Time-Data conundrum — I

A computational dogma

Running time of a learning algorithm increases with the size of the data.

A Time-Data conundrum — I

A computational dogma

Running time of a learning algorithm increases with the size of the data.

- Misaligned goals in the statistical and optimization disciplines

Discipline	Goal	Metric
Optimization	reaching numerical ϵ-accuracy	$\|\mathbf{x}^k - \mathbf{x}^\star\| \leq \epsilon$
Statistics	learning ε-accurate model	$\|\mathbf{x}^\star - \mathbf{x}^\natural\| \leq \varepsilon$

A Time-Data conundrum — I

A computational dogma

Running time of a learning algorithm increases with the size of the data.

- Misaligned goals in the statistical and optimization disciplines

Discipline	Goal	Metric
Optimization	reaching numerical ϵ-accuracy	$\|\mathbf{x}^k - \mathbf{x}^\star\| \leq \epsilon$
Statistics	learning ε-accurate model	$\|\mathbf{x}^\star - \mathbf{x}^\natural\| \leq \varepsilon$

- Main issue: ϵ **and** ε **are NOT the same but should be treated jointly!**

A Time-Data conundrum — II

A stylized formalization of the time-data tradeoff

The goals of optimization and statistical modeling are tightly connected:

$$\underbrace{\|\mathbf{x}^k - \mathbf{x}^\natural\|}_{\text{learning quality}} \leq \underbrace{\|\mathbf{x}^k - \mathbf{x}^\star\|}_{\epsilon\text{: needs ``time'' } t(k)} + \underbrace{\|\mathbf{x}^\star - \mathbf{x}^\natural\|}_{\varepsilon\text{: needs ``data''} n}$$

$\mathbf{x}^\natural$:	true model in $\mathbb{R}^p$
$\mathbf{x}^\star$:	statistical model estimate
$\mathbf{x}^k$:	numerical solution at iteration k

As the number of data samples n increases,

- ...with a fixed optimization formulation,

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_1 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{w}\|_2, \|\mathbf{x}\|_\infty \leq 1 \right\}$$

  - numerical methods take longer time t to reach ϵ-accuracy
    - e.g., per-iteration time to solve an $n \times 2p$ linear system
  - statistical model estimates ε become more precise when $\|\mathbf{w}\|_2 = \mathcal{O}(\sqrt{n})$
    - $\varepsilon = \dfrac{2\sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s}}{\sqrt{n} - \sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s} - \kappa} \|\mathbf{w}\|_2$, with probability $1 - 6\exp(-\kappa^2/26)$.

A Time-Data conundrum — II

A stylized formalization of the time-data tradeoff

The goals of optimization and statistical modeling are tightly connected:

$$\underbrace{\|\mathbf{x}^k - \mathbf{x}^\natural\|}_{\text{learning quality}} \leq \underbrace{\|\mathbf{x}^k - \mathbf{x}^\star\|}_{\epsilon\text{: needs "time" } t(k)} + \underbrace{\|\mathbf{x}^\star - \mathbf{x}^\natural\|}_{\varepsilon\text{: needs "data"} n} \leq \bar{\varepsilon}(t(k), n),$$

$\mathbf{x}^\natural$:	true model in $\mathbb{R}^p$
$\mathbf{x}^\star$:	statistical model estimate
$\mathbf{x}^k$:	numerical solution at iteration k
$\bar{\varepsilon}(t(k), n)$:	actual model precision at time $t(k)$ with n samples

As the number of data samples n increases,

- ...with a fixed optimization formulation,

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \|\mathbf{x}\|_1 : \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{w}\|_2, \|\mathbf{x}\|_\infty \leq 1 \right\}$$

  - numerical methods take longer time t to reach ϵ-accuracy
    - e.g., per-iteration time to solve an $n \times 2p$ linear system
  - statistical model estimates ε become more precise when $\|\mathbf{w}\|_2 = \mathcal{O}(\sqrt{n})$
    - $\varepsilon = \dfrac{2\sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s}}{\sqrt{n} - \sqrt{2s\log(\frac{p}{s}) + \frac{5}{4}s} - \kappa}\|\mathbf{w}\|_2$, with probability $1 - 6\exp(-\kappa^2/26)$.

"Time" effort has significant diminishing returns on ε in the underdetermined case*

(cf., [9, 5, 58, 7])

* "Data" effort also exhibits a similar behavior in the overdetermined case when a signal prior is used due to noise!

Data as a computational resource

A stylized formalization of the time-data tradeoff [7]

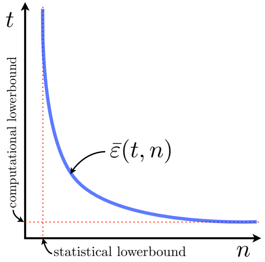
The goals of optimization and statistical modeling are tightly connected:

$$\underbrace{\|\mathbf{x}^{k(t)} - \mathbf{x}^\natural\|}_{\text{learning quality}} \leq \underbrace{\|\mathbf{x}^{k(t)} - \mathbf{x}^\star\|}_{\epsilon\text{: needs "time" } t} + \underbrace{\|\mathbf{x}^\star - \mathbf{x}^\natural\|}_{\varepsilon\text{: needs "data"} n} \leq \bar{\varepsilon}(t, n),$$

$\mathbf{x}^\natural$:	true model in $\mathbb{R}^p$
$\bar{\varepsilon}(t, n)$:	actual model precision at time t with n samples

Rest of the tutorial: **Time** $t(k)$ **aspects** (mostly)

- scalable algorithms
- stochastic approximations
- communication and synchronization aspects



Smooth unconstrained convex minimization

Problem (**Mathematical formulation**)

The unconstrained convex minimization problem is defined as:

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

- f *is a proper, closed and smooth convex function,* $-\infty < f^\star < +\infty$.
- *The solution set* $\mathcal{S}^\star := \{\mathbf{x}^\star \in \mathrm{dom}(f) : f(\mathbf{x}^\star) = f^\star\}$ *is nonempty.*

Example: Maximum likelihood estimation and M-estimators

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be unknown and* $b_1, ..., b_n$ *be i.i.d. samples of a random variable* B *with p.d.f.* $p_{\mathbf{x}^\natural}(b) \in \mathcal{P} := \{p_{\mathbf{x}}(b) : \mathbf{x} \in \mathbb{R}^p\}$*.*
Goal: estimate $\mathbf{x}^\natural$ *from* $b_1, \ldots, b_n$*.*

Optimization formulation (ML estimator)

$$\hat{\mathbf{x}}_{\mathsf{ML}} := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \ln\left[p_{\mathbf{x}}(b_i)\right] \right\} = \arg\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

Example: Maximum likelihood estimation and M-estimators

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ be unknown and $b_1, ..., b_n$ be i.i.d. samples of a random variable B with p.d.f. $p_{\mathbf{x}^\natural}(b) \in \mathcal{P} := \{p_{\mathbf{x}}(b) : \mathbf{x} \in \mathbb{R}^p\}$.
Goal: estimate $\mathbf{x}^\natural$ from $b_1, \ldots, b_n$.

Optimization formulation (ML estimator)

$$\hat{\mathbf{x}}_{\mathsf{ML}} := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \ln\left[p_{\mathbf{x}}(b_i)\right] \right\} = \arg\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

Theorem (Performance of the ML estimator [36, 65])

The random variable $\hat{\mathbf{x}}_{ML}$ satisfies

$$\lim_{n\to\infty} \sqrt{n}\,\mathbf{J}^{-1/2}\left(\hat{\mathbf{x}}_{ML} - \mathbf{x}^\natural\right) \overset{d}{=} Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where

$$\mathbf{J} := -\mathbb{E}\left[\nabla^2_{\mathbf{x}} \ln\left[p_{\mathbf{x}}(B)\right]\right]\Big|_{\mathbf{x}=\mathbf{x}^\natural}.$$

is the Fisher information matrix associated with one sample.

Example: Maximum likelihood estimation and M-estimators

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be unknown and* $b_1, ..., b_n$ *be i.i.d. samples of a random variable* B *with p.d.f.* $p_{\mathbf{x}^\natural}(b) \in \mathcal{P} := \{p_{\mathbf{x}}(b) : \mathbf{x} \in \mathbb{R}^p\}$.
Goal: estimate $\mathbf{x}^\natural$ *from* $b_1, \ldots, b_n$.

Optimization formulation (ML estimator)

$$\hat{\mathbf{x}}_{\mathsf{ML}} := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \ln\left[p_{\mathbf{x}}(b_i)\right] \right\} = \arg\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

Theorem (Performance of the ML estimator [36, 65])

The random variable $\hat{\mathbf{x}}_{ML}$ *satisfies*

$$\lim_{n\to\infty} \sqrt{n}\,\mathbf{J}^{-1/2}\left(\hat{\mathbf{x}}_{ML} - \mathbf{x}^\natural\right) \overset{d}{=} Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where

$$\mathbf{J} := -\mathbb{E}\left[\nabla^2_{\mathbf{x}} \ln\left[p_{\mathbf{x}}(B)\right]\right]\Big|_{\mathbf{x}=\mathbf{x}^\natural}.$$

is the Fisher information matrix associated with one sample. Roughly speaking,

$$\left\|\sqrt{n}\,\mathbf{J}^{-1/2}\left(\hat{\mathbf{x}}_{ML} - \mathbf{x}^\natural\right)\right\|_2^2 \sim \mathrm{Tr}\left(\mathbf{I}\right) = p \quad \Rightarrow \quad \boxed{\left\|\hat{\mathbf{x}}_{ML} - \mathbf{x}^\natural\right\|_2^2 = \mathcal{O}(p/n)}.$$

Example: Maximum likelihood estimation and M-estimators

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be unknown and* $b_1, ..., b_n$ *be i.i.d. samples of a random variable* B *with p.d.f.* $p_{\mathbf{x}^\natural}(b) \in \mathcal{P} := \{p_{\mathbf{x}}(b) : \mathbf{x} \in \mathbb{R}^p\}$*.*
Goal: estimate $\mathbf{x}^\natural$ *from* $b_1, \ldots, b_n$*.*

Optimization formulation (ML estimator)

$$\hat{\mathbf{x}}_{\mathsf{ML}} := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \ln\left[p_{\mathbf{x}}(b_i)\right] \right\} = \arg\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

Optimization formulation (M-estimator)

In general, we can replace the negative log-likelihoods by any appropriate, convex g_i's

$$\min_{x\in\mathcal{X}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} g_i(b_i;\mathbf{x})}_{f(\mathbf{x})}.$$

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

- Even when a closed-form solution exists, numerical accuracy may still be an issue.

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

- Even when a closed-form solution exists, numerical accuracy may still be an issue.
- We must be content with **approximately** optimal solutions.

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

- Even when a closed-form solution exists, numerical accuracy may still be an issue.
- We must be content with **approximately** optimal solutions.

Definition

We say that $\mathbf{x}^\star_\epsilon$ is ϵ-optimal in **objective value** if

$$f(\mathbf{x}^\star_\epsilon) - f^\star \leq \epsilon \ .$$

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

- Even when a closed-form solution exists, numerical accuracy may still be an issue.
- We must be content with **approximately** optimal solutions.

Definition

We say that $\mathbf{x}^\star_\epsilon$ is ϵ-optimal in **objective value** if

$$f(\mathbf{x}^\star_\epsilon) - f^\star \leq \epsilon \ .$$

Definition

We say that $\mathbf{x}^\star_\epsilon$ is ϵ-optimal in **sequence** if, for some norm $\|\cdot\|$,

$$\|\mathbf{x}^\star_\epsilon - \mathbf{x}^\star\| \leq \epsilon \ ,$$

Approximate vs. exact optimality

Is it possible to solve a convex optimization problem?

"In general, optimization problems are **unsolvable***"* - Y. Nesterov [46]

- Even when a closed-form solution exists, numerical accuracy may still be an issue.
- We must be content with **approximately** optimal solutions.

Definition

We say that $\mathbf{x}^\star_\epsilon$ is ϵ-optimal in **objective value** if

$$f(\mathbf{x}^\star_\epsilon) - f^\star \leq \epsilon \ .$$

Definition

We say that $\mathbf{x}^\star_\epsilon$ is ϵ-optimal in **sequence** if, for some norm $\|\cdot\|$,

$$\|\mathbf{x}^\star_\epsilon - \mathbf{x}^\star\| \leq \epsilon \ ,$$

- The latter approximation guarantee is considered stronger.

A gradient method

Lemma (First-order necessary optimality condition)

Let $\mathbf{x}^\star$ be a global minimum of a differentiable convex function f. Then, it holds that

$$\nabla f(\mathbf{x}^\star) = \mathbf{0}.$$

A gradient method

Lemma (First-order necessary optimality condition)

Let $\mathbf{x}^\star$ be a global minimum of a differentiable convex function f. Then, it holds that

$$\nabla f(\mathbf{x}^\star) = \mathbf{0}.$$

Fixed-point characterization

Multiply by -1 and add $\mathbf{x}^\star$ to both sides to obtain a fixed point condition,

$$\mathbf{x}^\star = \mathbf{x}^\star - \alpha \nabla f(\mathbf{x}^\star) \qquad \text{for all } 0 \neq \alpha \in \mathbb{R}$$

A gradient method

Lemma (First-order necessary optimality condition)

Let $\mathbf{x}^\star$ *be a global minimum of a differentiable convex function* f*. Then, it holds that*

$$\nabla f(\mathbf{x}^\star) = \mathbf{0}.$$

Fixed-point characterization

Multiply by -1 and add $\mathbf{x}^\star$ to both sides to obtain a fixed point condition,

$$\mathbf{x}^\star = \mathbf{x}^\star - \alpha \nabla f(\mathbf{x}^\star) \qquad \text{for all } 0 \neq \alpha \in \mathbb{R}$$

Gradient method

Choose a starting point $\mathbf{x}^0$ and iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k)$$

where α_k is a step-size to be chosen so that $\mathbf{x}^k$ converges to $\mathbf{x}^\star$.

When does the gradient method converge?

Lemma

Assume that

1. *There exists* $\mathbf{x}^\star \in$ *dom*(f) *such that* $\nabla f(\mathbf{x}^\star) = 0$.
2. *The mapping* $\psi(\mathbf{x}) = \mathbf{x} - \alpha \nabla f(\mathbf{x})$ *is contractive for some* α*: i.e., there exists* $\gamma \in [0, 1)$ *such that*

$$\|\psi(\mathbf{x}) - \psi(\mathbf{z})\| \leq \gamma \|\mathbf{x} - \mathbf{z}\| \quad \textit{for all } \mathbf{x}, \mathbf{z} \in \textit{dom}(f)$$

Then, for any starting point $\mathbf{x}^0 \in$ *dom*(f)*, the gradient method converges to* $\mathbf{x}^\star$.

When does the gradient method converge?

Lemma

Assume that

1. *There exists* $\mathbf{x}^\star \in$ *dom*(f) *such that* $\nabla f(\mathbf{x}^\star) = 0$.
2. *The mapping* $\psi(\mathbf{x}) = \mathbf{x} - \alpha\nabla f(\mathbf{x})$ *is contractive for some* α*: i.e., there exists* $\gamma \in [0, 1)$ *such that*

$$\|\psi(\mathbf{x}) - \psi(\mathbf{z})\| \leq \gamma\|\mathbf{x} - \mathbf{z}\| \quad \text{for all } \mathbf{x}, \mathbf{z} \in \text{dom}(f)$$

Then, for any starting point $\mathbf{x}^0 \in$ *dom*(f)*, the gradient method converges to* $\mathbf{x}^\star$.

Proof.

If we start the gradient method at $\mathbf{x}^0 \in \text{dom}(f)$, then we have

$$\begin{aligned}
\|\mathbf{x}^{k+1} - \mathbf{x}^\star\| &= \|\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k) - \mathbf{x}^\star\| \\
&= \|\psi(\mathbf{x}^k) - \psi(\mathbf{x}^\star)\| && (\nabla f(\mathbf{x}^\star) = 0) \\
&\leq \gamma\|\mathbf{x}^k - \mathbf{x}^\star\| && \text{(contraction)} \\
&\leq \gamma^{k+1}\|\mathbf{x}^0 - \mathbf{x}^\star\| \, .
\end{aligned}$$

We then have that the sequence $\{\mathbf{x}^k\}$ converges globally to $\mathbf{x}^\star$ at a **linear** rate.

□

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Convergence rates: the *"speed"* at which a sequence converges

- **sublinear:** if there exists $c > 0$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(k^{-c})$$

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Convergence rates: the *"speed"* at which a sequence converges

- **sublinear:** if there exists $c > 0$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(k^{-c})$$

- **linear:** if there exists $\alpha \in (0, 1)$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(\alpha^k)$$

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Convergence rates: the *"speed"* at which a sequence converges

- **sublinear:** if there exists $c > 0$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(k^{-c})$$

- **linear:** if there exists $\alpha \in (0, 1)$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(\alpha^k)$$

- **Q-linear:** if there exists a constant $r \in (0, 1)$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{u}^{k+1} - \mathbf{u}^\star\|}{\|\mathbf{u}^k - \mathbf{u}^\star\|} = r$$

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Convergence rates: the *"speed"* at which a sequence converges

- **sublinear:** if there exists $c > 0$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(k^{-c})$$

- **linear:** if there exists $\alpha \in (0, 1)$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(\alpha^k)$$

- **Q-linear:** if there exists a constant $r \in (0, 1)$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{u}^{k+1} - \mathbf{u}^\star\|}{\|\mathbf{u}^k - \mathbf{u}^\star\|} = r$$

- **superlinear:** If $r = 0$, we say that the sequence converges *superlinearly*.

Short (but important) detour: convergence rates

Definition (Convergence of a sequence)

The sequence $\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^k, ...$ converges to $\mathbf{u}^\star$ (denoted $\lim_{k\to\infty} \mathbf{u}^k = \mathbf{u}^\star$), if

$$\forall\ \varepsilon > 0, \exists\ K \in \mathbb{N} : k \geq K \Rightarrow \|\mathbf{u}^k - \mathbf{u}^\star\| \leq \varepsilon$$

Convergence rates: the *"speed"* at which a sequence converges

- **sublinear:** if there exists $c > 0$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(k^{-c})$$

- **linear:** if there exists $\alpha \in (0, 1)$ such that

$$\|\mathbf{u}^k - \mathbf{u}^\star\| = O(\alpha^k)$$

- **Q-linear:** if there exists a constant $r \in (0, 1)$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{u}^{k+1} - \mathbf{u}^\star\|}{\|\mathbf{u}^k - \mathbf{u}^\star\|} = r$$

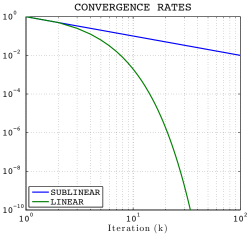
- **superlinear:** If $r = 0$, we say that the sequence converges *superlinearly*.
- **quadratic:** if there exists a constant $\mu > 0$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{u}^{k+1} - \mathbf{u}^\star\|}{\|\mathbf{u}^k - \mathbf{u}^\star\|^2} = \mu$$

Example: Convergence rates

Examples of sequences that all converge to $u^\star = 0$:

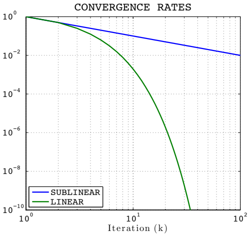
- Sublinear: $u^k = 1/k$
- Linear: $u^k = 0.5^k$

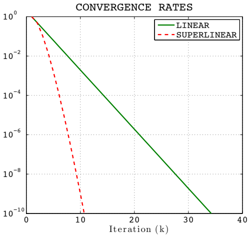


Example: Convergence rates

Examples of sequences that all converge to $u^\star = 0$:

- Sublinear: $u^k = 1/k$
- Linear: $u^k = 0.5^k$
- Superlinear: $u^k = k^{-k}$

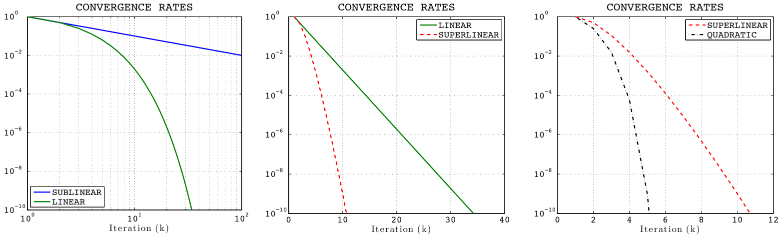




Example: Convergence rates

Examples of sequences that all converge to $u^\star = 0$:

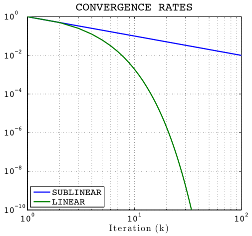
- Sublinear: $u^k = 1/k$
- Linear: $u^k = 0.5^k$
- Superlinear: $u^k = k^{-k}$
- Quadratic: $u^k = 0.5^{2^k}$

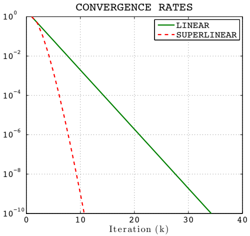


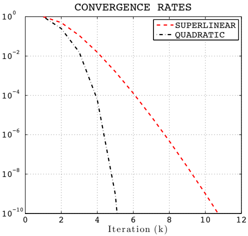
Example: Convergence rates

Examples of sequences that all converge to $u^\star = 0$:

- Sublinear: $u^k = 1/k$
- Linear: $u^k = 0.5^k$
- Superlinear: $u^k = k^{-k}$
- Quadratic: $u^k = 0.5^{2^k}$







Remark

For **unconstrained** convex minimization as in (1), we always have $f(\mathbf{x}^k) - f^\star \geq 0$. Hence, we do not need to use the absolute value when we show convergence results based on the objective value, such as $f(\mathbf{x}^k) - f^\star \leq O(1/k^2)$, which is sublinear.

Gradient descent methods

Definition

Gradient descent (GD) Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, update $\{\mathbf{x}^k\}_{k\geq 0}$ as

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) search direction.

Key question: how to choose α_k to have descent/contraction?

Gradient descent methods

Definition

Gradient descent (GD) Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, update $\{\mathbf{x}^k\}_{k\geq 0}$ as

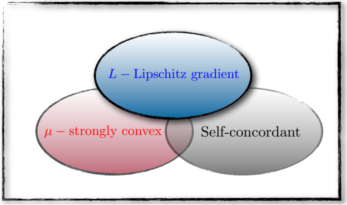
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) search direction.

Key question: how to choose α_k to have descent/contraction?

We need structure!

We use $\mathcal{F}$ to denote the class of smooth convex functions.
(The domain of each function will be apparent from the context.)



Gradient descent methods

Definition

Gradient descent (GD) Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, update $\{\mathbf{x}^k\}_{k\geq 0}$ as

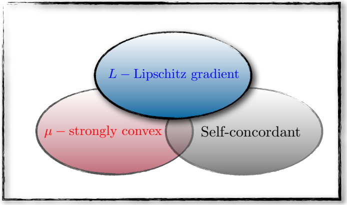
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) search direction.

Key question: how to choose α_k to have descent/contraction?

We need structure!

We use $\mathcal{F}$ to denote the class of smooth convex functions.
(The domain of each function will be apparent from the context.)



Next few slides: structural assumptions

L-Lipschitz gradient class of functions

Definition (L-Lipschitz gradient convex functions)

Let $f : \mathcal{Q} \to \mathbb{R}$ be differentiable and convex, i.e., $f \in \mathcal{F}^1(\mathcal{Q})$. Then, f has a Lipschitz gradient if there exists $L > 0$ (the Lipschitz constant) s.t.

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{Q}.$$

Proposition (L-Lipschitz gradient convex functions)

$f \in \mathcal{F}^1(\mathcal{Q})$ has L-Lipschitz gradient if and only if the following function is convex:

$$h(\mathbf{x}) = \frac{L}{2}\|\mathbf{x}\|_2^2 - f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{Q}.$$

L-Lipschitz gradient class of functions

Definition (L-Lipschitz gradient convex functions)

Let $f : \mathcal{Q} \to \mathbb{R}$ be differentiable and convex, i.e., $f \in \mathcal{F}^1(\mathcal{Q})$. Then, f has a Lipschitz gradient if there exists $L > 0$ (the Lipschitz constant) s.t.

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{Q}.$$

Proposition (L-Lipschitz gradient convex functions)

$f \in \mathcal{F}^1(\mathcal{Q})$ has L-Lipschitz gradient if and only if the following function is convex:

$$h(\mathbf{x}) = \frac{L}{2}\|\mathbf{x}\|_2^2 - f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{Q}.$$

Definition (Class of 2-nd order Lipschitz functions)

The class of twice continuously differentiable functions f on $\mathcal{Q}$ with Lipschitz continuous Hessian is denoted as $\mathcal{F}_L^{2,2}(\mathcal{Q})$ (with $2 \to 2$ denoting the spectral norm)

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\|_{2\to 2} \leq L\|\mathbf{x} - \mathbf{y}\|_2, \quad \forall \mathbf{x}, \mathbf{y} \in Q,$$

- $\mathcal{F}_L^{l,m}$: functions that are l-times differentiable with m-th order Lipschitz property.

Example: Logistic regression

Problem (Logistic regression [34])

Given a sample vector $\mathbf{a}_i \in \mathbb{R}^p$ and a binary class label $b_i \in \{-1, +1\}$ ($i = 1, \ldots, n$), we define the conditional probability of b_i given $\mathbf{a}_i$ as:

$$\mathbb{P}(b_i | \mathbf{a}_i, \mathbf{x}^\natural, \mu) \propto 1/(1 + e^{-b_i(\langle \mathbf{x}^\natural, \mathbf{a}_i \rangle + \mu)}),$$

where $\mathbf{x}^\natural \in \mathbb{R}^p$ is some true weight vector, $\mu \in \mathbb{R}$ is called the intercept. How to estimate $\mathbf{x}^\natural$ given the sample vectors, the binary labels, and μ?

Example: Logistic regression

Problem (Logistic regression [34])

Given a sample vector $\mathbf{a}_i \in \mathbb{R}^p$ and a binary class label $b_i \in \{-1, +1\}$ ($i = 1, \ldots, n$), we define the conditional probability of b_i given $\mathbf{a}_i$ as:

$$\mathbb{P}(b_i | \mathbf{a}_i, \mathbf{x}^\natural, \mu) \propto 1/(1 + e^{-b_i(\langle \mathbf{x}^\natural, \mathbf{a}_i \rangle + \mu)}),$$

where $\mathbf{x}^\natural \in \mathbb{R}^p$ is some true weight vector, $\mu \in \mathbb{R}$ is called the intercept. How to estimate $\mathbf{x}^\natural$ given the sample vectors, the binary labels, and μ?

Optimization formulation

$$\min_{\mathbf{x} \in \mathbb{R}^p} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-b_i(\mathbf{a}_i^T \mathbf{x} + \mu)))}_{f(\mathbf{x})}$$

Example: Logistic regression

Problem (Logistic regression [34])

Given a sample vector $\mathbf{a}_i \in \mathbb{R}^p$ and a binary class label $b_i \in \{-1, +1\}$ ($i = 1, \ldots, n$), we define the conditional probability of b_i given $\mathbf{a}_i$ as:

$$\mathbb{P}(b_i|\mathbf{a}_i, \mathbf{x}^\natural, \mu) \propto 1/(1 + e^{-b_i(\langle \mathbf{x}^\natural, \mathbf{a}_i\rangle + \mu)}),$$

where $\mathbf{x}^\natural \in \mathbb{R}^p$ is some true weight vector, $\mu \in \mathbb{R}$ is called the intercept. How to estimate $\mathbf{x}^\natural$ given the sample vectors, the binary labels, and μ?

Optimization formulation

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-b_i(\mathbf{a}_i^T\mathbf{x} + \mu)))}_{f(\mathbf{x})}$$

Structural properties

Let $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]^T$ (design matrix), then $f \in \mathcal{F}_L^{2,1}$, with $L = \frac{1}{4}\|\mathbf{A}^T\mathbf{A}\|$

Strong convexity

Definition

A convex function $f : \mathcal{Q} \to \mathbb{R}$ is said to be μ-strongly convex if

$$h(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|_2^2$$

is convex, where μ is called the strong convexity parameter.

Strong convexity

Definition

A convex function $f : \mathcal{Q} \to \mathbb{R}$ is said to be μ-strongly convex if

$$h(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|_2^2$$

is convex, where μ is called the strong convexity parameter.

- The class of k-differentiable μ-strongly functions is denoted as $\mathcal{F}_{\mu}^{k}(\mathcal{Q})$.

Strong convexity

Definition

A convex function $f : \mathcal{Q} \to \mathbb{R}$ is said to be μ-strongly convex if

$$h(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|_2^2$$

is convex, where μ is called the strong convexity parameter.

- The class of k-differentiable μ-strongly functions is denoted as $\mathcal{F}_\mu^k(\mathcal{Q})$.
- Non-smooth functions can be μ-strongly convex: e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{x}\|_2^2$.

Strong convexity

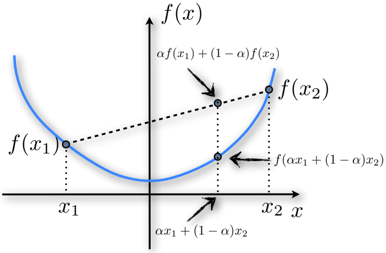
Definition

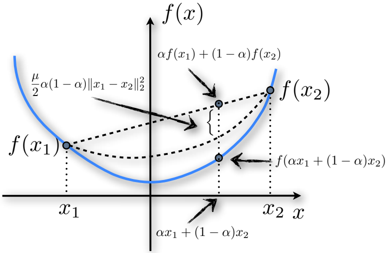
A convex function $f : \mathcal{Q} \to \mathbb{R}$ is said to be μ-strongly convex if

$$h(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|_2^2$$

is convex, where μ is called the strong convexity parameter.

- The class of k-differentiable μ-strongly functions is denoted as $\mathcal{F}_{\mu}^{k}(\mathcal{Q})$.
- Non-smooth functions can be μ-strongly convex: e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{x}\|_2^2$.





Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *and* $\mathbf{A} \in \mathbb{R}^{n \times p}$ *(full column rank). Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{A}$ *and*

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ *denotes unknown noise.*

Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *and* $\mathbf{A} \in \mathbb{R}^{n\times p}$ *(full column rank). Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{A}$ *and*

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ *denotes unknown noise.*

Optimization formulation (Least-squares estimator)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})}.$$

Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *and* $\mathbf{A} \in \mathbb{R}^{n \times p}$ *(full column rank). Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{A}$ *and*

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ *denotes unknown noise.*

Optimization formulation (Least-squares estimator)

$$\min_{\mathbf{x} \in \mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})} .$$

Structural properties

- $\nabla f(\mathbf{x}) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, and $\nabla^2 f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}$.

Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ and $\mathbf{A} \in \mathbb{R}^{n\times p}$ (full column rank). Goal: estimate $\mathbf{x}^\natural$, given $\mathbf{A}$ and

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ denotes unknown noise.

Optimization formulation (Least-squares estimator)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})}.$$

Structural properties

- $\nabla f(\mathbf{x}) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, and $\nabla^2 f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}$.
- $\lambda_p \mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq \lambda_1 \mathbf{I}$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *and* $\mathbf{A} \in \mathbb{R}^{n\times p}$ *(full column rank). Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{A}$ *and*

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ *denotes unknown noise.*

Optimization formulation (Least-squares estimator)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})}.$$

Structural properties

- $\nabla f(\mathbf{x}) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, and $\nabla^2 f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}$.
- $\lambda_p \mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq \lambda_1 \mathbf{I}$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.
- It follows that $L = \lambda_1$ and $\mu = \lambda_p$. If $\lambda_p > 0$, then $f \in \mathcal{F}_{L,\mu}^{2,1}$, otherwise $f \in \mathcal{F}_L^{2,1}$.

Example: Least-squares estimation

Problem

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *and* $\mathbf{A} \in \mathbb{R}^{n\times p}$ *(full column rank). Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{A}$ *and*

$$\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w},$$

where $\mathbf{w}$ *denotes unknown noise.*

Optimization formulation (Least-squares estimator)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})}.$$

Structural properties

- $\nabla f(\mathbf{x}) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, and $\nabla^2 f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}$.
- $\lambda_p\mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq \lambda_1\mathbf{I}$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.
- It follows that $L = \lambda_1$ and $\mu = \lambda_p$. If $\lambda_p > 0$, then $f \in \mathcal{F}^{2,1}_{L,\mu}$, otherwise $f \in \mathcal{F}^{2,1}_{L}$.
- Since $\mathrm{rank}(\mathbf{A}^T\mathbf{A}) \leq \min\{n, p\}$, if $n < p$, then $\lambda_p = 0$.

Self-concordant functions

Proposition

A continuously differentiable function $f : \mathcal{Q} \to \mathbb{R}, \mathcal{Q} \subseteq \mathbb{R}^p$ is both μ-strongly and L-Lipschitz ($0 < \mu \leq L$) convex if and only if

$$\forall\ \mathbf{x}, \mathbf{y} \in \mathcal{Q}, \quad \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2.$$

Self-concordant functions

Proposition

A continuously differentiable function $f : \mathcal{Q} \to \mathbb{R}, \mathcal{Q} \subseteq \mathbb{R}^p$ is both μ-strongly and L-Lipschitz $(0 < \mu \leq L)$ convex if and only if

$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{Q}, \quad \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2.$$

Definition (Self-concordant functions)

A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be self-concordant with parameter $M \geq 0$, if

$$|\varphi'''(t)| \leq M\varphi''(t)^{3/2} \quad \text{, where} \quad \varphi(t) := f(\mathbf{x} + t\mathbf{v}),$$

for all $t \in \mathbb{R}$, $\mathbf{x} \in \mathrm{dom} f$ and $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{x} + t\mathbf{v} \in \mathrm{dom} f$.

If $M = 2$, then f is said to be *standard* self-concordant, i.e., $f \in \mathcal{F}_2$.

Self-concordant functions

Proposition

A continuously differentiable function $f : \mathcal{Q} \to \mathbb{R}, \mathcal{Q} \subseteq \mathbb{R}^p$ is both μ-strongly and L-Lipschitz ($0 < \mu \leq L$) convex if and only if

$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{Q}, \quad \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2.$$

Definition (Self-concordant functions)

A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be self-concordant with parameter $M \geq 0$, if

$$|\varphi'''(t)| \leq M\varphi''(t)^{3/2} \quad \text{, where} \quad \varphi(t) := f(\mathbf{x} + t\mathbf{v}),$$

for all $t \in \mathbb{R}$, $\mathbf{x} \in \mathrm{dom}f$ and $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{x} + t\mathbf{v} \in \mathrm{dom}f$.
If $M = 2$, then f is said to be *standard* self-concordant, i.e., $f \in \mathcal{F}_2$.

- This structure provides **local** lower and upper bounds

  $$\omega(\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}) \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \omega_*(\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}), \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathrm{dom}(f),$$

  where the second inequality locally holds for $\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} < 1$.

Self-concordant functions

Proposition

A continuously differentiable function $f : \mathcal{Q} \to \mathbb{R}, \mathcal{Q} \subseteq \mathbb{R}^p$ is both μ-strongly and L-Lipschitz ($0 < \mu \leq L$) convex if and only if

$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{Q}, \quad \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2.$$

Definition (Self-concordant functions)

A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be self-concordant with parameter $M \geq 0$, if

$$|\varphi'''(t)| \leq M\varphi''(t)^{3/2} \text{ , where } \varphi(t) := f(\mathbf{x} + t\mathbf{v}),$$

for all $t \in \mathbb{R}$, $\mathbf{x} \in \mathrm{dom} f$ and $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{x} + t\mathbf{v} \in \mathrm{dom} f$.
If $M = 2$, then f is said to be *standard* self-concordant, i.e., $f \in \mathcal{F}_2$.

- This structure provides **local** lower and upper bounds

  $$\omega(\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}) \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \omega_*(\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}), \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathrm{dom}(f),$$

  where the second inequality locally holds for $\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} < 1$.
  - ω-functions: $\omega(\tau) = \tau - \ln(1 + \tau)$ and $\omega_*(\tau) = -\tau - \ln(1 - \tau)$ for $\tau \in (0, 1]$.

Self-concordant functions

Proposition

A continuously differentiable function $f : \mathcal{Q} \to \mathbb{R}, \mathcal{Q} \subseteq \mathbb{R}^p$ is both μ-strongly and L-Lipschitz ($0 < \mu \leq L$) convex if and only if

$$\forall\, \mathbf{x}, \mathbf{y} \in \mathcal{Q}, \quad \frac{\mu}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y}-\mathbf{x} \rangle \leq \frac{L}{2}\|\mathbf{y}-\mathbf{x}\|_2^2.$$

Definition (Self-concordant functions)

A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be self-concordant with parameter $M \geq 0$, if

$$|\varphi'''(t)| \leq M\varphi''(t)^{3/2} \quad \text{, where} \quad \varphi(t) := f(\mathbf{x} + t\mathbf{v}),$$

for all $t \in \mathbb{R}$, $\mathbf{x} \in \mathrm{dom} f$ and $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{x} + t\mathbf{v} \in \mathrm{dom} f$.

If $M = 2$, then f is said to be *standard* self-concordant, i.e., $f \in \mathcal{F}_2$.

- This structure provides **local** lower and upper bounds

  $$\omega(\|\mathbf{y}-\mathbf{x}\|_{\mathbf{x}}) \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y}-\mathbf{x} \rangle \leq \omega_*(\|\mathbf{y}-\mathbf{x}\|_{\mathbf{x}}), \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathrm{dom}(f),$$

  where the second inequality locally holds for $\|\mathbf{y}-\mathbf{x}\|_{\mathbf{x}} < 1$.
  - ω-functions: $\omega(\tau) = \tau - \ln(1+\tau)$ and $\omega_*(\tau) = -\tau - \ln(1-\tau)$ for $\tau \in (0, 1]$.
  - Local norm: $\|\mathbf{y}\|_{\mathbf{x}} := \left[\mathbf{y}^T \nabla^2 f(\mathbf{x}) \mathbf{y}\right]^{1/2}$.

Example: Poisson imaging

Problem (Poisson imaging [23])

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be an unknown vector and* $b_1, \ldots, b_n$ *be samples of*

$$B_i \sim \textit{Poisson}\Big(\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle\Big), \quad \textit{for } i = 1, ..., n,$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *are given. Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *and* $b_1, \ldots, b_n$*.*

[a] `http://lions.epfl.ch/scopt`

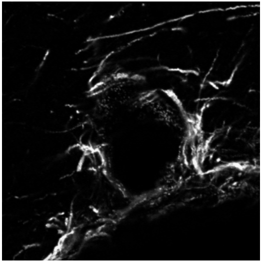
Example: Poisson imaging

Problem (Poisson imaging [23])

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be an unknown vector and* $b_1, \ldots, b_n$ *be samples of*

$$B_i \sim \textit{Poisson}\Big(\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle\Big), \quad \textit{for } i = 1, ..., n,$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *are given.* *Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *and* $b_1, \ldots, b_n$*.*



Example: confocal (as many other photon-limited) images are Poissonian[a]

[a] http://lions.epfl.ch/scopt

Example: Poisson imaging

Problem (Poisson imaging [23])

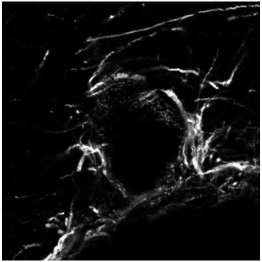
Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be an unknown vector and* $b_1, \ldots, b_n$ *be samples of*

$$B_i \sim \textit{Poisson}\Big(\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle\Big), \quad \textit{for } i = 1, ..., n,$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *are given. Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *and* $b_1, \ldots, b_n$*.*

Optimization formulation

$$\hat{\mathbf{x}}_{\mathrm{ML}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left[\langle \mathbf{a}_i, \mathbf{x}\rangle - b_i \ln\left(\langle \mathbf{a}_i, \mathbf{x}\rangle\right)\right]}_{f(\mathbf{x})}.$$



Example: confocal (as many other photon-limited) images are Poissonian[a]

[a] http://lions.epfl.ch/scopt

Example: Poisson imaging

Problem (Poisson imaging [23])

Let $\mathbf{x}^\natural \in \mathbb{R}^p$ *be an unknown vector and* $b_1, \ldots, b_n$ *be samples of*

$$B_i \sim \textit{Poisson}\Big(\langle \mathbf{a}_i, \mathbf{x}^\natural \rangle\Big), \quad \textit{for } i = 1, ..., n,$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *are given.* *Goal: estimate* $\mathbf{x}^\natural$*, given* $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *and* $b_1, \ldots, b_n$*.*

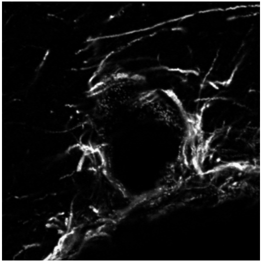
Optimization formulation

$$\hat{\mathbf{x}}_{\mathrm{ML}} \in \arg\min_{\mathbf{x} \in \mathbb{R}^p} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left[\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \ln\left(\langle \mathbf{a}_i, \mathbf{x} \rangle \right) \right]}_{f(\mathbf{x})}.$$

Structural properties [61]

- $f \in \mathcal{F}_2$ is self-concordant with domain $\mathcal{Q} = \{\mathbf{x} : \langle \mathbf{a}_i, \mathbf{x} \rangle \geq 0, i = 1, \ldots, n\}$ and self-concordancy parameter

$$M = 2 \max \left\{ \frac{1}{\sqrt{b_i}} \ : \ b_i > 0, \ i = 1, \ldots, n \right\}$$



Example: confocal (as many other photon-limited) images are Poissonian[a]

[a] http://lions.epfl.ch/scopt

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \text{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.
- α_k can be determined by a line-search procedure:

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.
- α_k can be determined by a line-search procedure:
  1. **Exact line search**: $\alpha_k := \arg\min_{\alpha > 0} f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k))$.

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.
- α_k can be determined by a line-search procedure:
  1. **Exact line search**: $\alpha_k := \arg\min_{\alpha > 0} f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k))$.
  2. **Back-tracking line search** with Armijo-Goldstein's condition:

$$f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)) \leq f(\mathbf{x}^k) - c\alpha \|\nabla f(\mathbf{x}^k)\|^2, \quad c \in (0, 1/2].$$

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.
- α_k can be determined by a line-search procedure:
  1. **Exact line search**: $\alpha_k := \arg\min_{\alpha > 0} f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k))$.
  2. **Back-tracking line search** with Armijo-Goldstein's condition:

$$f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)) \leq f(\mathbf{x}^k) - c\alpha \|\nabla f(\mathbf{x}^k)\|^2, \quad c \in (0, 1/2].$$

Case 2: If $f \in \mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k \leq \frac{2}{L+\mu}$. The optimal choice is $\alpha_k := \frac{2}{L+\mu}$.

Back to gradient descent methods

Gradient descent (GD) algorithm

Starting from $\mathbf{x}^0 \in \mathrm{dom}(f)$, produce the sequence $\mathbf{x}^1, ..., \mathbf{x}^k, ...$ according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) = \mathbf{x}^k + \alpha_k \mathbf{p}^k.$$

Notice that $\mathbf{p}^k := -\nabla f(\mathbf{x}^k)$ is the steepest descent (anti-gradient) direction.
Key question: how do we choose α_k to have descent/contraction?

Step-size selection

Case 1: If $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, then:

- We can choose $0 < \alpha_k < \frac{2}{L}$. The optimal choice is $\alpha_k := \frac{1}{L}$.
- α_k can be determined by a line-search procedure:
  1. **Exact line search**: $\alpha_k := \arg\min_{\alpha > 0} f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k))$.
  2. **Back-tracking line search** with Armijo-Goldstein's condition:

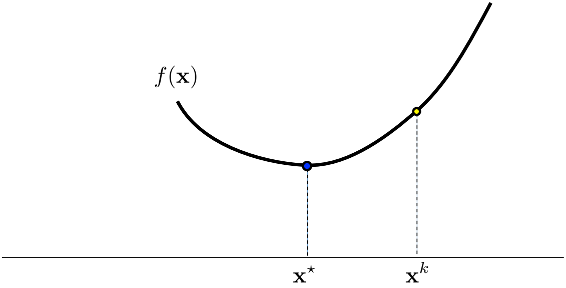
$$f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)) \leq f(\mathbf{x}^k) - c\alpha \|\nabla f(\mathbf{x}^k)\|^2, \quad c \in (0, 1/2].$$

Case 2: If $f \in \mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^p)$, then:

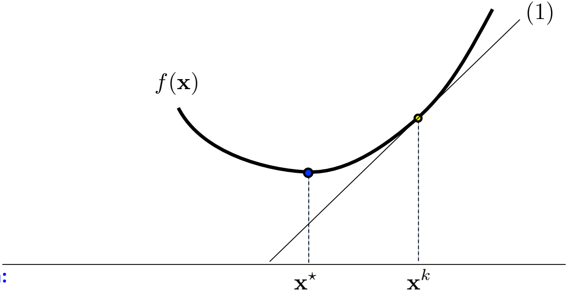
- We can choose $0 < \alpha_k \leq \frac{2}{L+\mu}$. The optimal choice is $\alpha_k := \frac{2}{L+\mu}$.

Case 3: If $f \in \mathcal{F}_2(\mathcal{Q})$, then, a bit more complicated (more later).

Gradient descent methods: geometrical intuition



Gradient descent methods: geometrical intuition



Structure in optimization:

(1) $\quad f(\mathbf{x}) \geq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle$

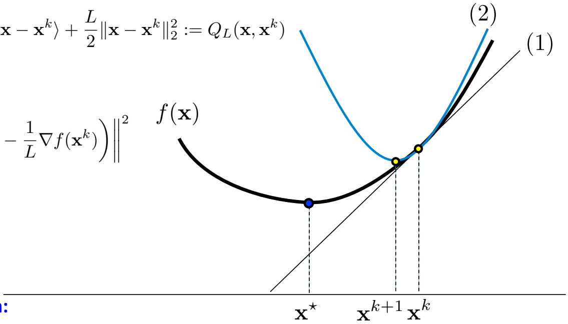
Gradient descent methods: geometrical intuition

Majorize:

$$f(\mathbf{x}) \le f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 := Q_L(\mathbf{x}, \mathbf{x}^k)$$

Minimize:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} Q_L(\mathbf{x}, \mathbf{x}^k) \\
&= \arg\min_{\mathbf{x}} \left\| \mathbf{x} - \left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k) \right) \right\|^2 \\
&= \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)
\end{aligned}$$



Structure in optimization:

(1) $\quad f(\mathbf{x}) \ge f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle$

(2) $\quad f(\mathbf{x}) \le f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

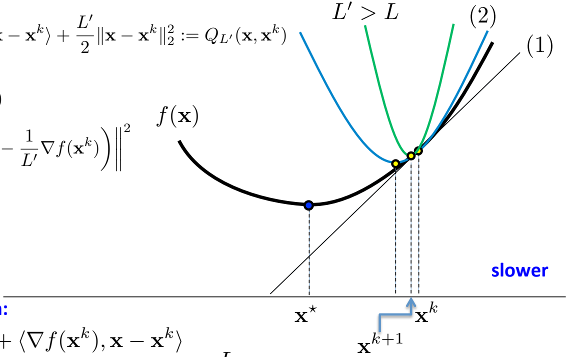
Gradient descent methods: geometrical intuition

Majorize:

$$f(\mathbf{x}) \le f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L'}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 := Q_{L'}(\mathbf{x}, \mathbf{x}^k)$$

Minimize:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} Q_{L'}(\mathbf{x}, \mathbf{x}^k) \\ &= \arg\min_{\mathbf{x}} \left\| \mathbf{x} - \left(\mathbf{x}^k - \frac{1}{L'}\nabla f(\mathbf{x}^k) \right) \right\|^2 \\ &= \mathbf{x}^k - \frac{1}{L'}\nabla f(\mathbf{x}^k)\end{aligned}$$



Structure in optimization:

(1) $\qquad f(\mathbf{x}) \ge f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle$

(2) $\qquad f(\mathbf{x}) \le f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

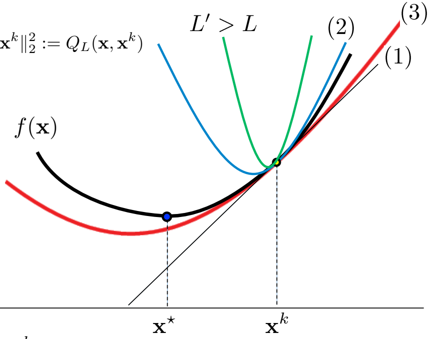
Gradient descent methods: geometrical intuition

Majorize:

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 := Q_L(\mathbf{x}, \mathbf{x}^k)$$

Minimize:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} Q_L(\mathbf{x}, \mathbf{x}^k) \\
&= \arg\min_{\mathbf{x}} \left\| \mathbf{x} - \left(\mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k) \right) \right\|^2 \\
&= \mathbf{x}^k - \frac{1}{L}\nabla f(\mathbf{x}^k)
\end{aligned}$$



Structure in optimization:

(1) $\quad f(\mathbf{x}) \geq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle$

(2) $\quad f(\mathbf{x}) \leq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

(3) $\quad f(\mathbf{x}) \geq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

Convergence rate of gradient descent

Theorem

$$f \in \mathcal{F}_L^{2,1}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(\mathbf{x}^\star) \leq \frac{2L}{k+4}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2$$

$$f \in \mathcal{F}_{L,\mu}^{2,1}, \quad \alpha = \frac{2}{L+\mu}: \qquad \|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{L-\mu}{L+\mu}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

$$f \in \mathcal{F}_{L,\mu}^{2,1}, \quad \alpha = \frac{1}{L}: \qquad \|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

Note that $\frac{L-\mu}{L+\mu} = \frac{\kappa-1}{\kappa+1}$, where $\kappa := \frac{L}{\mu}$ is the condition number of $\nabla^2 f$.

Convergence rate of gradient descent

Theorem

$$f \in \mathcal{F}_L^{2,1}, \quad \alpha = \frac{1}{L}: \qquad f(\mathbf{x}^k) - f(\mathbf{x}^\star) \leq \frac{2L}{k+4}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2$$

$$f \in \mathcal{F}_{L,\mu}^{2,1}, \quad \alpha = \frac{2}{L+\mu}: \qquad \|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{L-\mu}{L+\mu}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

$$f \in \mathcal{F}_{L,\mu}^{2,1}, \quad \alpha = \frac{1}{L}: \qquad \|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

Note that $\frac{L-\mu}{L+\mu} = \frac{\kappa-1}{\kappa+1}$, where $\kappa := \frac{L}{\mu}$ is the condition number of $\nabla^2 f$.

Remarks

- **Assumption:** Lipschitz gradient. **Result:** convergence rate in **objective values**.
- **Assumption:** Strong convexity. **Result:** convergence rate in **sequence** of the iterates and in **objective values**.
- Note that the suboptimal step-size choice $\alpha = \frac{1}{L}$ adapts to the strongly convex case (i.e., it features a linear rate vs. the standard sublinear rate).

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

Remarks

- $f \in \mathcal{F}_{L,\mu}^{2,1}$ with:

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

Remarks

- $f \in \mathcal{F}_{L,\mu}^{2,1}$ with:
    - $L = \lambda_p(\mathbf{A}^T\mathbf{A}) + \rho;$
    - $\mu = \lambda_1(\mathbf{A}^T\mathbf{A}) + \rho;$

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n\times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

Remarks

- $f \in \mathcal{F}_{L,\mu}^{2,1}$ with:
  - $L = \lambda_p(\mathbf{A}^T\mathbf{A}) + \rho$;
  - $\mu = \lambda_1(\mathbf{A}^T\mathbf{A}) + \rho$;
  - where $\lambda_1 \leq \ldots \leq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

Remarks

- $f \in \mathcal{F}_{L,\mu}^{2,1}$ with:
  - $L = \lambda_p(\mathbf{A}^T\mathbf{A}) + \rho$;
  - $\mu = \lambda_1(\mathbf{A}^T\mathbf{A}) + \rho$;
  - where $\lambda_1 \leq \ldots \leq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.
- The ratio $\kappa = \frac{L}{\mu}$ decreases as ρ increases, leading to faster linear convergence.

Example: Ridge regression

Optimization formulation

- Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{b} \in \mathbb{R}^n$ given by $\mathbf{b} = \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^n$ is some noise.
- A classical estimator of $\mathbf{x}^\natural$, known as ridge regression, is

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) := \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{x}\|_2^2.$$

where $\rho \geq 0$ is a regularization parameter

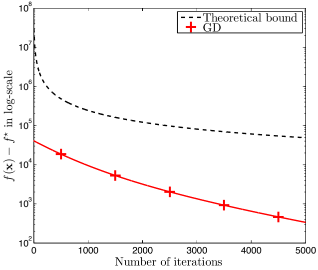
Remarks

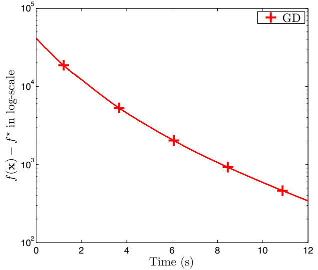
- $f \in \mathcal{F}_{L,\mu}^{2,1}$ with:
  - $L = \lambda_p(\mathbf{A}^T\mathbf{A}) + \rho$;
  - $\mu = \lambda_1(\mathbf{A}^T\mathbf{A}) + \rho$;
  - where $\lambda_1 \leq \ldots \leq \lambda_p$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.
- The ratio $\kappa = \frac{L}{\mu}$ decreases as ρ increases, leading to faster linear convergence.
- Note that if $n < p$ and $\rho = 0$, we have $\mu = 0$, hence $f \in \mathcal{F}_L^{2,1}$ and we can expect only $\mathcal{O}(1/k)$ convergence from the gradient descent method.

Example: Ridge regression

Case 1:

$n = 500, p = 2000, \rho = 0$

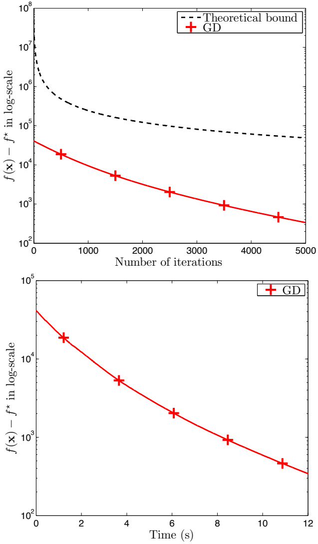




Example: Ridge regression

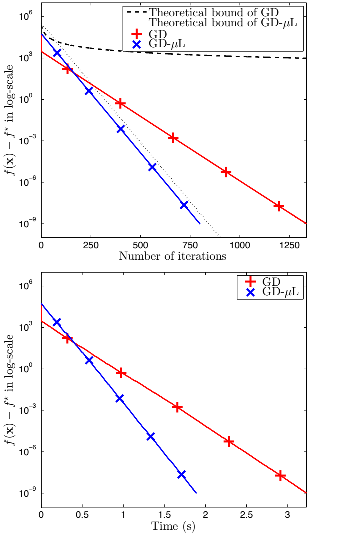
Case 1:

$n = 500, p = 2000, \rho = 0$



Case 2:

$n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



Theoretical worst-case complexity lower bounds

What is the **best** achievable **rate** for a **first-order** method (i.e., using function and gradient values, no higher-order information)?

Theoretical worst-case complexity lower bounds

What is the **best** achievable **rate** for a **first-order** method (i.e., using function and gradient values, no higher-order information)?

Theorem [46]

It is possible to construct a function $f \in \mathcal{F}_L^{\infty,1}$ (smooth and with L-Lipschitz gradient), for which **any** first-order method satisfies

$$f(\mathbf{x}^k) - f(\mathbf{x}^\star) \geq \frac{3L}{32(k+1)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \quad \text{for all } k \leq (p-1)/2$$

Theoretical worst-case complexity lower bounds

What is the **best** achievable **rate** for a **first-order** method (i.e., using function and gradient values, no higher-order information)?

Theorem [46]

It is possible to construct a function $f \in \mathcal{F}_L^{\infty,1}$ (smooth and with L-Lipschitz gradient), for which **any** first-order method satisfies

$$f(\mathbf{x}^k) - f(\mathbf{x}^\star) \geq \frac{3L}{32(k+1)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \quad \text{for all } k \leq (p-1)/2$$

Theorem [46]

It is possible to construct a function in $f \in \mathcal{F}_{L,\mu}^{\infty,1}$ (smooth, μ-strongly convex, and with L-Lipschitz gradient), for which **any** first order method satisfies

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \geq \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

Theoretical worst-case complexity lower bounds

What is the **best** achievable **rate** for a **first-order** method (i.e., using function and gradient values, no higher-order information)?

Theorem [46]

It is possible to construct a function $f \in \mathcal{F}_L^{\infty,1}$ (smooth and with L-Lipschitz gradient), for which **any** first-order method satisfies

$$f(\mathbf{x}^k) - f(\mathbf{x}^\star) \geq \frac{3L}{32(k+1)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \quad \text{for all } k \leq (p-1)/2$$

Theorem [46]

It is possible to construct a function in $f \in \mathcal{F}_{L,\mu}^{\infty,1}$ (smooth, μ-strongly convex, and with L-Lipschitz gradient), for which **any** first order method satisfies

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \geq \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$$

Gradient descent is $O(1/k)$ for $\mathcal{F}_L^{\infty,1}$ and it is slower for $\mathcal{F}_{L,\mu}^{\infty,1}$, hence it does not achieve the lower bounds!

Accelerated Gradient Algorithm

Problem

Is it possible to design optimal first-order methods with convergence rates matching the theoretical lower bounds?

Accelerated Gradient Algorithm

Problem

Is it possible to design optimal first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient (AG) methods achieve optimal convergence rates at a negligible increase in the computational cost.

Accelerated Gradient Algorithm

Problem

Is it possible to design optimal first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient (AG) methods achieve optimal convergence rates at a negligible increase in the computational cost.

Accelerated Gradient algorithm for $\mathcal{F}_L^{1,1}$ (AG-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} \mathbf{x}^{k+1} & = \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} & = (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated Gradient Algorithm

Problem

Is it possible to design optimal first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient (AG) methods achieve optimal convergence rates at a negligible increase in the computational cost.

Accelerated Gradient algorithm for $\mathcal{F}_L^{1,1}$ (AG-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} &= (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated Gradient algorithm for $\mathcal{F}_{L,\mu}^{1,1}$ (AG-μL)

1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \mathrm{dom}(f)$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} &= \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ \mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \gamma(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

where $\gamma = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.

Accelerated Gradient Algorithm

Problem

Is it possible to design optimal first-order methods with convergence rates matching the theoretical lower bounds?

Solution [Nesterov's accelerated scheme]

Accelerated Gradient (AG) methods achieve optimal convergence rates at a negligible increase in the computational cost.

Accelerated Gradient algorithm for $\mathcal{F}_L^{1,1}$ (AG-L)

1. Set $\mathbf{x}^0 = \mathbf{y}^0 \in \text{dom}(f)$ and $t_0 := 1$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} & = \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ t_{k+1} & = (1 + \sqrt{4t_k^2 + 1})/2 \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} + \frac{(t_k - 1)}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

Accelerated Gradient algorithm for $\mathcal{F}_{L,\mu}^{1,1}$ (AG-μL)

1. Choose $\mathbf{x}^0 = \mathbf{y}^0 \in \text{dom}(f)$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{x}^{k+1} & = \mathbf{y}^k - \frac{1}{L}\nabla f(\mathbf{y}^k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} + \gamma(\mathbf{x}^{k+1} - \mathbf{x}^k) \end{cases}$$

where $\gamma = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.

NOTE: AG is not monotone, but the cost-per-iteration is essentially the same as GD.

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ or $\mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ *or* $\mathcal{F}_{L,\mu}^{1,1}$, *the sequence* $\{\mathbf{x}^k\}_{k\geq 0}$ *generated by* **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2,\ \forall k \geq 0.$$

AG-L is **optimal** *for* $\mathcal{F}_L^{1,1}$ *but NOT for* $\mathcal{F}_{L,\mu}^{1,1}$ *with known* μ*!*

Theorem (f is strongly convex with Lipschitz gradient)

If $f \in \mathcal{F}_{L,\mu}^{1,1}$, *the sequence* $\{\mathbf{x}^k\}_{k\geq 0}$ *generated by* **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2,\ \forall k \geq 0$$

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2,\ \forall k \geq 0.$$

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ or $\mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

AG-L is **optimal** *for $\mathcal{F}_L^{1,1}$ but NOT for $\mathcal{F}_{L,\mu}^{1,1}$ with known μ!*

Theorem (f is strongly convex with Lipschitz gradient)

If $f \in \mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0$$

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2, \ \forall k \geq 0.$$

- AG-L's iterates are not guarantee to converge.

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ or $\mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

AG-L is ***optimal*** *for $\mathcal{F}_L^{1,1}$ but NOT for $\mathcal{F}_{L,\mu}^{1,1}$ with known μ!*

Theorem (f is strongly convex with Lipschitz gradient)

If $f \in \mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0$$

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2, \ \forall k \geq 0.$$

- AG-L's iterates are not guarantee to converge.
- AG-L does not have a **linear** convergence rate for $\mathcal{F}_{L,\mu}^{1,1}$.

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ or $\mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2,\ \forall k \geq 0.$$

AG-L is **optimal** *for $\mathcal{F}_L^{1,1}$ but NOT for $\mathcal{F}_{L,\mu}^{1,1}$ with known μ!*

Theorem (f is strongly convex with Lipschitz gradient)

If $f \in \mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2,\ \forall k \geq 0$$

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2,\ \forall k \geq 0.$$

- AG-L's iterates are not guarantee to converge.
- AG-L does not have a **linear** convergence rate for $\mathcal{F}_{L,\mu}^{1,1}$.
- AG-μL does, but needs to know μ.

Global convergence of AG [46]

Theorem (f is convex with Lipschitz gradient)

If $f \in \mathcal{F}_L^{1,1}$ or $\mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AG-L** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq \frac{4L}{(k+2)^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0.$$

AG-L is **optimal** *for $\mathcal{F}_L^{1,1}$ but NOT for $\mathcal{F}_{L,\mu}^{1,1}$ with known μ!*

Theorem (f is strongly convex with Lipschitz gradient)

If $f \in \mathcal{F}_{L,\mu}^{1,1}$, the sequence $\{\mathbf{x}^k\}_{k\geq 0}$ generated by **AGD-μL** *satisfies*

$$f(\mathbf{x}^k) - f^\star \leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2, \ \forall k \geq 0$$

$$\|\mathbf{x}^k - \mathbf{x}^\star\|_2 \leq \sqrt{\frac{2L}{\mu}}\left(1 - \sqrt{\frac{\mu}{L}}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2, \ \forall k \geq 0.$$

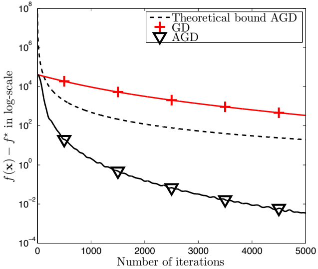
- AG-L's iterates are not guarantee to converge.
- AG-L does not have a **linear** convergence rate for $\mathcal{F}_{L,\mu}^{1,1}$.
- AG-μL does, but needs to know μ.

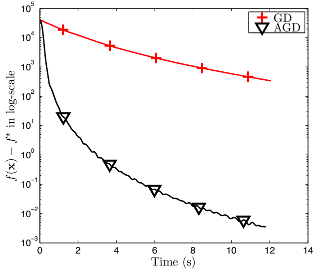
AGD achieves the iteration lowerbound within a constant!

Example: Ridge regression

Case 1:

$n = 500, p = 2000, \rho = 0$

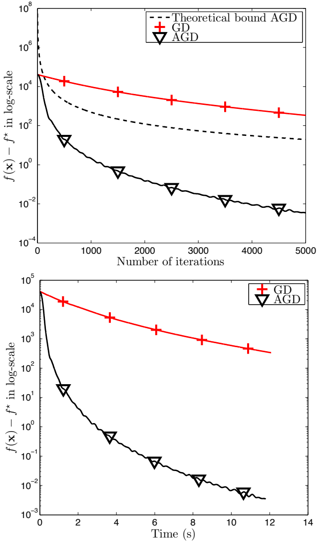




Example: Ridge regression

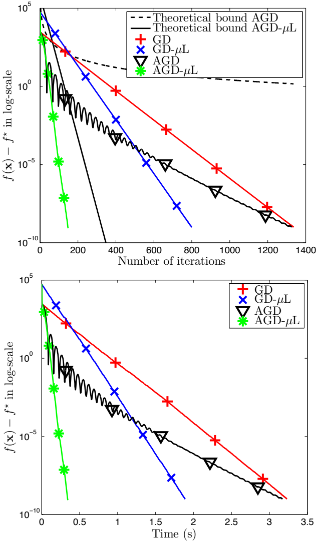
Case 1:

$n = 500, p = 2000, \rho = 0$



Case 2:

$n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

When do we need a line-search procedure?

We can use a line-search procedure for both GD and AGD when

- L is **known** but it is expensive to evaluate;
- The global constant L usually does not capture the local behavior of f or it is **unknown**;

Enhancements

Two enhancements

1. Line-search for estimating L for both GD and AGD.
2. Restart strategies for AGD.

When do we need a line-search procedure?

We can use a line-search procedure for both GD and AGD when

- L is **known** but it is expensive to evaluate;
- The global constant L usually does not capture the local behavior of f or it is **unknown**;

Line-search

At each iteration, we try to find a constant L_k that satisfies:

$$f(\mathbf{x}^{k+1}) \leq Q_{L_k}(\mathbf{x}^{k+1}, \mathbf{y}^k) := f(\mathbf{y}^k) + \langle \nabla f(\mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{y}^k \rangle + \frac{L_k}{2} \|\mathbf{x}^{k+1} - \mathbf{y}^k\|_2^2.$$

Here: $L_0 > 0$ is given (e.g., $L_0 := c\frac{\|\nabla f(\mathbf{x}^1) - \nabla f(\mathbf{x}^0)\|_2}{\|\mathbf{x}^1 - \mathbf{x}^0\|_2}$) for $c \in (0, 1]$.

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.
- AG is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.
- AG is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AG has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.
- AG is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AG has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.
- A **restart strategy** tries to reset this momentum whenever we observe high periodic behavior. We often use function values but other strategies are possible.

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.
- AG is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AG has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.
- A **restart strategy** tries to reset this momentum whenever we observe high periodic behavior. We often use function values but other strategies are possible.

Two restart strategies

1. **O'Donoghue - Candes's strategy [51]:** There are at least three options: Restart with fixed number of iterations, restart based on objective values, and restart based on a gradient condition.

Enhancements

Why do we need a restart strategy?

- AG-μL requires knowledge of μ and AG-L does not have optimal convergence for strongly convex f.
- AG is non-monotonic (i.e., $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k)$ is not always satisfied).
- AG has a periodic behavior, where the momentum depends on the local condition number $\kappa = L/\mu$.
- A **restart strategy** tries to reset this momentum whenever we observe high periodic behavior. We often use function values but other strategies are possible.

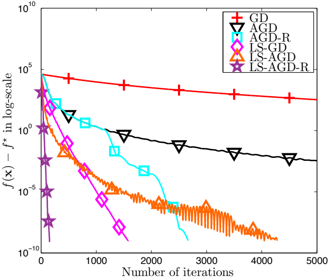
Two restart strategies

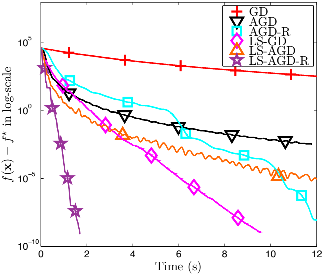
1. **O'Donoghue - Candes's strategy [51]:** There are at least three options: Restart with fixed number of iterations, restart based on objective values, and restart based on a gradient condition.
2. **Giselsson-Boyd's strategy [27]:** Do not require $t_k = 1$ and do not necessary require function evaluations.

Example: Ridge regression

Case 1:

$n = 500, p = 2000, \rho = 0$

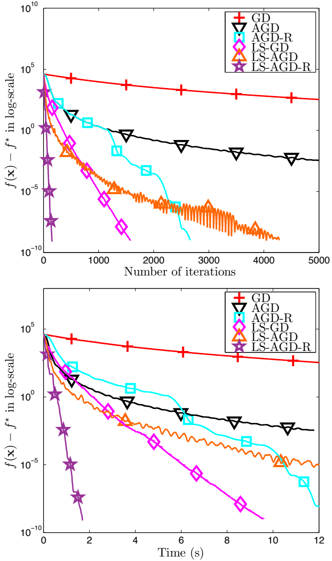




Example: Ridge regression

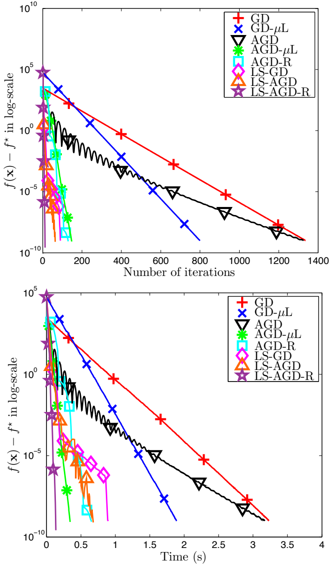
Case 1:

$n = 500, p = 2000, \rho = 0$

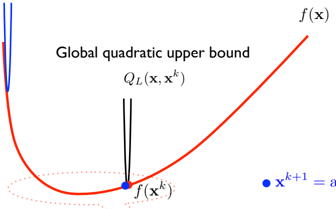


Case 2:

$n = 500, p = 2000, \rho = 0.01\lambda_p(\mathbf{A}^T\mathbf{A})$



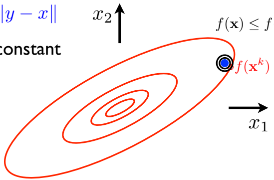
How can we better adapt to the local geometry?



- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

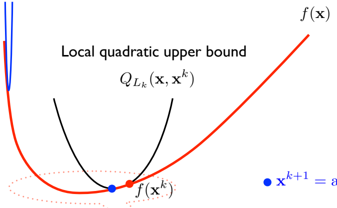
$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$

L is a global worst-case constant



$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

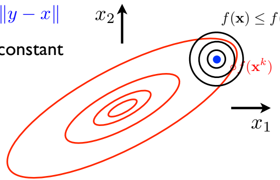
How can we better adapt to the local geometry?



- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$

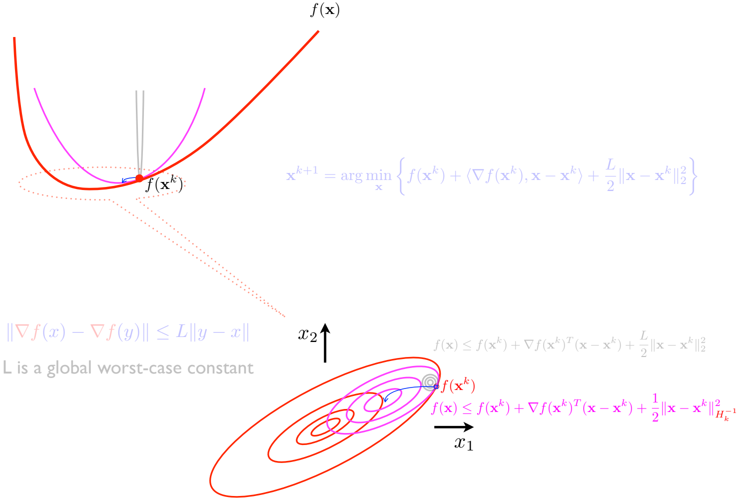
L is a global worst-case constant



$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

applies only locally

How can we better adapt to the local geometry?



Variable metric gradient descent algorithm

Variable metric gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}^k), \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.
3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Variable metric gradient descent algorithm

Variable metric gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots,$ perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k), \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Common choices of the variable metric $\mathbf{H}_k$

- $\mathbf{H}_k := \lambda_k \mathbf{I}$ $\Longrightarrow$ gradient descent method.
- $\mathbf{H}_k := \mathbf{D}_k$ (a positive diagonal matrix) $\Longrightarrow$ scaled gradient descent method.
- $\mathbf{H}_k := \nabla^2 f(\mathbf{x}^k)$ $\Longrightarrow$ Newton method.
- $\mathbf{H}_k \approx \nabla^2 f(\mathbf{x}^k)$ $\Longrightarrow$ quasi-Newton method.

Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**, :

$$\nabla^2 f(\mathbf{x}^k) \mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**, :

$$\nabla^2 f(\mathbf{x}^k) \mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

- A unit step-size $\alpha_k = 1$ can be chosen near convergence:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k) \, .$$

Newton method

- Fast (local) convergence but expensive per iteration cost
- **Useful** when warm-started near a solution

Local quadratic approximation using the Hessian

- Obtain a local quadratic approximation using the second-order Taylor series approximation to $f(\mathbf{x}^k + \mathbf{p})$:

$$f(\mathbf{x}^k + \mathbf{p}) \approx f(\mathbf{x}^k) + \langle \mathbf{p}, \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{p}, \nabla^2 f(\mathbf{x}^k) \mathbf{p} \rangle$$

- The Newton direction is the vector $\mathbf{p}^k$ that minimizes $f(\mathbf{x}^k + \mathbf{p})$; assuming the Hessian $\nabla^2 f_k$ to be **positive definite**, :

$$\nabla^2 f(\mathbf{x}^k) \mathbf{p}^k = -\nabla f(\mathbf{x}^k) \quad \Leftrightarrow \quad \mathbf{p}^k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$$

- A unit step-size $\alpha_k = 1$ can be chosen near convergence:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k) \ .$$

Remark

- For $f \in \mathcal{F}_L^{2,1}$ but $f \notin \mathcal{F}_{L,\mu}^{2,1}$, the Hessian may not always be positive definite.

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k \nabla f(\mathbf{x}^k).$$

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k \nabla f(\mathbf{x}^k).$$

- Matrix $\mathbf{H}_k$, or its inverse $\mathbf{B}_k$, undergoes low-rank updates:
  - Rank 1 or 2 updates: famous Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.
  - Limited memory BFGS (L-BFGS).

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k \nabla f(\mathbf{x}^k).$$

- Matrix $\mathbf{H}_k$, or its inverse $\mathbf{B}_k$, undergoes low-rank updates:
  - Rank 1 or 2 updates: famous Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.
  - Limited memory BFGS (L-BFGS).
- Line-search: The step-size α_k is chosen to satisfy the **Wolfe conditions**:

$$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) \leq f(\mathbf{x}^k) + c_1 \alpha_k \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(sufficient decrease)}$$

$$\langle \nabla f(\mathbf{x}^k + \alpha_k \mathbf{p}^k), \mathbf{p}^k \rangle \geq c_2 \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(curvature condition)}$$

with $0 < c_1 < c_2 < 1$. For quasi-Newton methods, we usually use $c_1 = 0.1$.

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k \nabla f(\mathbf{x}^k).$$

- Matrix $\mathbf{H}_k$, or its inverse $\mathbf{B}_k$, undergoes low-rank updates:
  - Rank 1 or 2 updates: famous Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.
  - Limited memory BFGS (L-BFGS).
- Line-search: The step-size α_k is chosen to satisfy the **Wolfe conditions**:

$$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) \le f(\mathbf{x}^k) + c_1 \alpha_k \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(sufficient decrease)}$$
$$\langle \nabla f(\mathbf{x}^k + \alpha_k \mathbf{p}^k), \mathbf{p}^k \rangle \ge c_2 \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(curvature condition)}$$

  with $0 < c_1 < c_2 < 1$. For quasi-Newton methods, we usually use $c_1 = 0.1$.
- Convergence is guaranteed under the Dennis & Moré condition [17].

Quasi-Newton methods

Quasi-Newton methods use an approximate Hessian oracle and can be more scalable.

- Useful for $f(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$ with $n \gg p$.

Main ingredients

Quasi-Newton direction:

$$\mathbf{p}^k = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^k) = -\mathbf{B}_k \nabla f(\mathbf{x}^k).$$

- Matrix $\mathbf{H}_k$, or its inverse $\mathbf{B}_k$, undergoes low-rank updates:
  - Rank 1 or 2 updates: famous Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.
  - Limited memory BFGS (L-BFGS).
- Line-search: The step-size α_k is chosen to satisfy the **Wolfe conditions**:

$$f(\mathbf{x}^k + \alpha_k \mathbf{p}^k) \leq f(\mathbf{x}^k) + c_1 \alpha_k \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(sufficient decrease)}$$

$$\langle \nabla f(\mathbf{x}^k + \alpha_k \mathbf{p}^k), \mathbf{p}^k \rangle \geq c_2 \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \qquad \text{(curvature condition)}$$

  with $0 < c_1 < c_2 < 1$. For quasi-Newton methods, we usually use $c_1 = 0.1$.
- Convergence is guaranteed under the Dennis & Moré condition [17].
- For more details on quasi-Newton methods, see Nocedal&Wright's book [50].

Example: Logistic regression

Problem (Logistic regression)

Given $\mathbf{A} \in \{0,1\}^{n\times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$, *solve:*

$$f^\star := \min_{\mathbf{x},\beta} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} \log\Big(1 + \exp\Big(-\mathbf{b}_j(\mathbf{a}_j^T\mathbf{x} + \beta)\Big)\Big) \right\}.$$

Example: Logistic regression

Problem (Logistic regression)

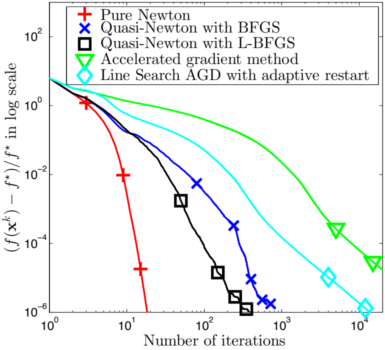
Given $\mathbf{A} \in \{0,1\}^{n\times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$*, solve:*

$$f^\star := \min_{\mathbf{x},\beta}\left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} \log\Big(1+\exp\big(-\mathbf{b}_j(\mathbf{a}_j^T\mathbf{x}+\beta)\big)\Big)\right\}.$$

Real data

- Real data: `w5a` with $n = 9888$ data points, $p = 300$ features
- Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

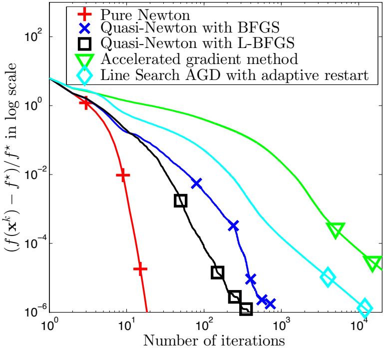
Example: Logistic regression - numerical results

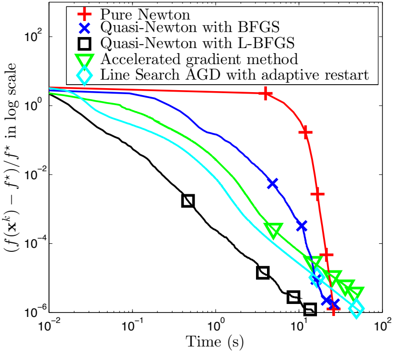


Parameters

- For BFGS, L-BFGS and Newton's method: maximum number of iterations 200, tolerance 10^{-6}. L-BFGS memory $m = 50$.
- For accelerated gradient method: maximum number of iterations 20000, tolerance 10^{-6}.
- Ground truth: Get a high accuracy approximation of $\mathbf{x}^\star$ and $f^\star$ by applying Newton's method for 200 iterations.

Example: Logistic regression - numerical results





Parameters

- For BFGS, L-BFGS and Newton's method: maximum number of iterations 200, tolerance 10^{-6}. L-BFGS memory $m = 50$.
- For accelerated gradient method: maximum number of iterations 20000, tolerance 10^{-6}.
- Ground truth: Get a high accuracy approximation of $\mathbf{x}^\star$ and $f^\star$ by applying Newton's method for 200 iterations.

Performance of optimization algorithms

Time-to-reach ϵ accuracy

`time-to-reach` ϵ `= number of iterations to reach` ϵ $\times$ `time-per-iteration`

The **speed** of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an ϵ-optimal solution.
- **Per-iteration time** depends on the information oracles, implementation, and the computational platform.

Performance of optimization algorithms

Time-to-reach ϵ accuracy

`time-to-reach` ϵ `= number of iterations to reach` ϵ $\times$ `time-per-iteration`

The **speed** of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an ϵ-optimal solution.
- **Per-iteration time** depends on the information oracles, implementation, and the computational platform.

In general, convergence rate and per-iteration time are inversely proportional.

Performance of optimization algorithms

Time-to-reach ϵ accuracy

`time-to-reach` ϵ `= number of iterations to reach` ϵ $\times$ `time-per-iteration`

The **speed** of numerical solutions depends on two factors:

- **Convergence rate** determines the number of iterations needed to obtain an ϵ-optimal solution.
- **Per-iteration time** depends on the information oracles, implementation, and the computational platform.

In general, convergence rate and per-iteration time are inversely proportional.
Finding the **fastest** algorithm is tricky! A non-exhaustive illustration:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system
Self-concordant, smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system

Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system
Self-concordant, smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system

Accelerated gradient descent:

$$
\begin{aligned}
\mathbf{x}^{k+1} &= \mathbf{y}^k - \alpha \nabla f(\mathbf{y}^k) \\
\mathbf{y}^{k+1} &= \mathbf{x}^{k+1} + \gamma_{k+1}(\mathbf{x}^{k+1} - \mathbf{x}^k).
\end{aligned}
$$

for some proper choice of α and γ_{k+1}.

Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system
Self-concordant, smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system

Main computations of the Quasi-Newton method

$$\mathbf{p}^k = -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}^k) ,$$

where $\mathbf{B}_k^{-1}$ is updated at each iteration by adding a rank-2 matrix.

Performance of optimization algorithms

A non-exhaustive comparison:

Assumptions on f	Algorithm	Convergence rate	Iteration complexity
Lipschitz-gradient $f \in \mathcal{F}_L^{2,1}(\mathbb{R}^p)$	Gradient descent	Sublinear $(1/k)$	One gradient
	Accelerated GD	Sublinear $(1/k^2)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system
Strongly convex, smooth $f \in \mathcal{F}_{L,\mu}^{2,1}(\mathbb{R}^p)$	Gradient descent	Linear (e^{-k})	One gradient
	Accelerated GD	Linear (e^{-k})	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Linear (e^{-k}), Quadratic	One gradient, one linear system
Self-concordant, smooth	Gradient descent	Sublinear $(1/k)$	One gradient
	Quasi-Newton	Superlinear	One gradient, rank-2 update
	Newton method	Sublinear $(1/k)$, Quadratic	One gradient, one linear system

The main computational bottleneck of **the Newton method** is the following

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k).$$

We can use **conjugate gradient algorithms** to efficiently solve this linear system.

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- *f and g are both proper, closed, and convex.*

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- f *and* g *are both proper, closed, and convex.*
- *dom*(F) := *dom*$(f) \cap$ *dom*$(g) \neq \emptyset$ *and* $-\infty < F^\star < +\infty$.

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- *f and g are both proper, closed, and convex.*
- *dom(F) := dom$(f) \cap$ dom$(g) \neq \emptyset$ and $-\infty < F^\star < +\infty$.*
- *The solution set $\mathcal{S}^\star := \{\mathbf{x}^\star \in \text{dom}(F) : F(\mathbf{x}^\star) = F^\star\}$ is nonempty.*

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- *f and g are both proper, closed, and convex.*
- *dom(F) := dom$(f) \cap$ dom$(g) \neq \emptyset$ and $-\infty < F^\star < +\infty$.*
- *The solution set $\mathcal{S}^\star := \{\mathbf{x}^\star \in$ dom$(F) : F(\mathbf{x}^\star) = F^\star\}$ is nonempty.*

Two remarks

- Nonsmoothness: At least one of the two functions f and g is **nonsmooth**

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\right\}$$

- *f and g are both proper, closed, and convex.*
- *dom(F) := dom$(f) \cap$ dom$(g) \neq \emptyset$ and $-\infty < F^\star < +\infty$.*
- *The solution set $\mathcal{S}^\star := \{\mathbf{x}^\star \in$ dom$(F) : F(\mathbf{x}^\star) = F^\star\}$ is nonempty.*

Two remarks

- Nonsmoothness: At least one of the two functions f and g is **nonsmooth**
  - General nonsmooth convex optimization methods (e.g., classical subgradient methods, level, or bundle methods) lack efficiency and numerical robustness.
    - Require $\mathcal{O}(\epsilon^{-2})$ iterations to reach a point $\mathbf{x}^\star_\epsilon$ such that $F(\mathbf{x}^\star_\epsilon) - F^\star \leq \epsilon$. Hence, to reach $\mathbf{x}^\star_{0.01}$ such that $F(\mathbf{x}^\star_{0.01}) - F^\star \leq 0.01$, we need $\mathcal{O}(10^4)$ iterations.

Composite convex minimization

Problem (Unconstrained composite convex minimization)

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- *f and g are both proper, closed, and convex.*
- *dom(F) := dom$(f) \cap$ dom$(g) \neq \emptyset$ and $-\infty < F^\star < +\infty$.*
- *The solution set $\mathcal{S}^\star := \{\mathbf{x}^\star \in \text{dom}(F) : F(\mathbf{x}^\star) = F^\star\}$ is nonempty.*

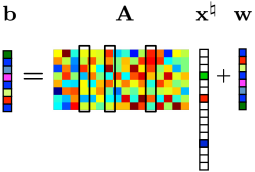
Two remarks

- Nonsmoothness: At least one of the two functions f and g is **nonsmooth**
  - General nonsmooth convex optimization methods (e.g., classical subgradient methods, level, or bundle methods) lack efficiency and numerical robustness.
    - Require $\mathcal{O}(\epsilon^{-2})$ iterations to reach a point $\mathbf{x}^\star_\epsilon$ such that $F(\mathbf{x}^\star_\epsilon) - F^\star \leq \epsilon$. Hence, to reach $\mathbf{x}^\star_{0.01}$ such that $F(\mathbf{x}^\star_{0.01}) - F^\star \leq 0.01$, we need $\mathcal{O}(10^4)$ iterations.
- Generality: it covers a wider range of problems than smooth unconstrained problems. E.g. when handling regularized M-estimation,
  - f is a loss function, a data fidelity, or negative log-likelihood function.
  - g is a regularizer, encouraging structure and/or constraints in the solution.

Example 1: Sparse regression in generalized linear models (GLMs)

Problem (Sparse regression in GLM)

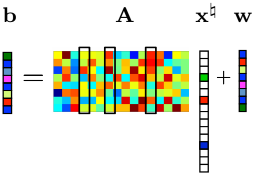
Our goal is to estimate $\mathbf{x}^\natural \in \mathbb{R}^p$ given $\{b_i\}_{i=1}^n$ and $\{\mathbf{a}_i\}_{i=1}^n$, knowing that the likelihood function at y_i given $\mathbf{a}_i$ and $\mathbf{x}^\natural$ is given by $\mathcal{L}(b_i; \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle)$, and that $\mathbf{x}^\natural$ is sparse.



Example 1: Sparse regression in generalized linear models (GLMs)

Problem (Sparse regression in GLM)

Our goal is to estimate $\mathbf{x}^\natural \in \mathbb{R}^p$ *given* $\{b_i\}_{i=1}^n$ *and* $\{\mathbf{a}_i\}_{i=1}^n$, *knowing that the likelihood function at* y_i *given* $\mathbf{a}_i$ *and* $\mathbf{x}^\natural$ *is given by* $\mathcal{L}(b_i; \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle)$, *and that* $\mathbf{x}^\natural$ *is sparse.*



Optimization formulation

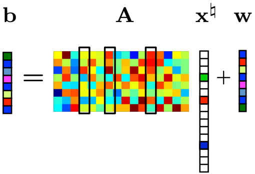
$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \underbrace{-\sum_{i=1}^n \log \mathcal{L}(b_i; \langle \mathbf{a}_i, \mathbf{x}\rangle)}_{f(\mathbf{x})} + \underbrace{\rho_n \|\mathbf{x}\|_1}_{g(\mathbf{x})} \right\}$$

where $\rho_n > 0$ is a parameter which controls the strength of sparsity regularization.

Example 1: Sparse regression in generalized linear models (GLMs)

Problem (Sparse regression in GLM)

Our goal is to estimate $\mathbf{x}^\natural \in \mathbb{R}^p$ given $\{b_i\}_{i=1}^n$ and $\{\mathbf{a}_i\}_{i=1}^n$, knowing that the likelihood function at y_i given $\mathbf{a}_i$ and $\mathbf{x}^\natural$ is given by $\mathcal{L}(b_i; \langle \mathbf{a}_i, \mathbf{x}^\natural \rangle)$, and that $\mathbf{x}^\natural$ is sparse.



Optimization formulation

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \underbrace{-\sum_{i=1}^n \log \mathcal{L}(b_i; \langle \mathbf{a}_i, \mathbf{x}\rangle)}_{f(\mathbf{x})} + \underbrace{\rho_n \|\mathbf{x}\|_1}_{g(\mathbf{x})} \right\}$$

where $\rho_n > 0$ is a parameter which controls the strength of sparsity regularization.

Theorem (cf. [38, 39, 43] for details)

Under some technical conditions, there exists $\{\rho_i\}_{i=1}^\infty$ such that with high probability,

$$\left\| \mathbf{x}^\star - \mathbf{x}^\natural \right\|_2^2 = \mathcal{O}\left(\frac{s \log p}{n}\right), \quad \operatorname{supp} \mathbf{x}^\star = \operatorname{supp} \mathbf{x}^\natural.$$

Recall ML: $\left\| \mathbf{x}_{ML} - \mathbf{x}^\natural \right\|_2^2 = \mathcal{O}(p/n)$.

Example 2: Image processing

Problem (Imaging denoising/deblurring)

Our goal is to obtain a clean image $\mathbf{x}$ *given "dirty" observations* $\mathbf{b} \in \mathbb{R}^{n \times 1}$ *via* $\mathbf{b} = \mathcal{A}(\mathbf{x}) + \mathbf{w}$, *where* $\mathcal{A}$ *is a linear operator, which, e.g., captures camera blur as well as image subsampling, and* $\mathbf{w}$ *models perturbations, such as Gaussian or Poisson noise.*

Optimization formulation

$$\text{Gaussian :} \quad \min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \Big\{ \underbrace{(1/2)\|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|_2^2}_{f(\mathbf{x})} + \underbrace{\rho\|\mathbf{x}\|_{\mathrm{TV}}}_{g(\mathbf{x})} \Big\}$$

$$\text{Poisson :} \quad \min_{\mathbf{x} \in \mathbb{R}^{n \times p}} \Big\{ \underbrace{\frac{1}{n}\sum_{i=1}^{n} \left[\langle \mathbf{a}_i, \mathbf{x}\rangle - b_i \ln\left(\langle \mathbf{a}_i, \mathbf{x}\rangle\right)\right]}_{f(\mathbf{x})} + \underbrace{\rho\|\mathbf{x}\|_{\mathrm{TV}}}_{g(\mathbf{x})} \Big\}$$

where $\rho > 0$ is a regularization parameter and $\|\cdot\|_{\mathrm{TV}}$ is the total variation (TV) norm:

$$\|\mathbf{x}\|_{\mathrm{TV}} := \begin{cases} \sum_{i,j} |\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j}| + |\mathbf{x}_{i+1,j} - \mathbf{x}_{i,j}| & \text{anisotropic case,} \\ \sum_{i,j} \sqrt{|\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j}|^2 + |\mathbf{x}_{i+1,j} - \mathbf{x}_{i,j}|^2} & \text{isotropic case} \end{cases}$$

Proximal-gradient method: A quadratic majorization perspective

Definition (Moreau proximal operator [40, 57])

Let $g \in \mathcal{F}(\mathbb{R}^p)$. The proximal operator (or prox-operator) of g is defined as:

$$\mathrm{prox}_g(\mathbf{x}) \equiv \arg\min_{\mathbf{y}\in\mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 \right\}.$$

Proximal-gradient method: A quadratic majorization perspective

Definition (Moreau proximal operator [40, 57])

Let $g \in \mathcal{F}(\mathbb{R}^p)$. The proximal operator (or prox-operator) of g is defined as:

$$\mathrm{prox}_g(\mathbf{x}) \equiv \arg\min_{\mathbf{y}\in\mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 \right\}.$$

Quadratic upper bound for f

For $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, we have, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$

$$f(\mathbf{x}) \leq f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 := Q_L(\mathbf{x},\mathbf{y})$$

Proximal-gradient method: A quadratic majorization perspective

Definition (Moreau proximal operator [40, 57])

Let $g \in \mathcal{F}(\mathbb{R}^p)$. The proximal operator (or prox-operator) of g is defined as:

$$\mathrm{prox}_g(\mathbf{x}) \equiv \arg\min_{\mathbf{y} \in \mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \right\}.$$

Quadratic upper bound for f

For $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, we have, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$

$$f(\mathbf{x}) \leq f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 := Q_L(\mathbf{x}, \mathbf{y})$$

Quadratic *majorizer* for $f + g$ [24]

Of course, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$,

$$f(\mathbf{x}) \leq Q_L(\mathbf{x}, \mathbf{y}) \quad \Rightarrow \quad f(\mathbf{x}) + g(\mathbf{x}) \leq Q_L(\mathbf{x}, \mathbf{y}) + g(\mathbf{x}) := P_L(\mathbf{x}, \mathbf{y})$$

Proximal-gradient method: A quadratic majorization perspective

Definition (Moreau proximal operator [40, 57])

Let $g \in \mathcal{F}(\mathbb{R}^p)$. The proximal operator (or prox-operator) of g is defined as:

$$\mathrm{prox}_g(\mathbf{x}) \equiv \arg\min_{\mathbf{y}\in\mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 \right\}.$$

Quadratic upper bound for f

For $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, we have, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$

$$f(\mathbf{x}) \le f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 := Q_L(\mathbf{x},\mathbf{y})$$

Quadratic *majorizer* for $f + g$ [24]

Of course, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$,

$$f(\mathbf{x}) \le Q_L(\mathbf{x},\mathbf{y}) \quad \Rightarrow \quad f(\mathbf{x}) + g(\mathbf{x}) \le Q_L(\mathbf{x},\mathbf{y}) + g(\mathbf{x}) := P_L(\mathbf{x},\mathbf{y})$$

Proximal-gradient from the majorize-minimize perspective [24]

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} P_L(\mathbf{x},\mathbf{x}^k)$$

Proximal-gradient method: A quadratic majorization perspective

Definition (Moreau proximal operator [40, 57])

Let $g \in \mathcal{F}(\mathbb{R}^p)$. The proximal operator (or prox-operator) of g is defined as:

$$\mathrm{prox}_g(\mathbf{x}) \equiv \arg\min_{\mathbf{y} \in \mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \right\}.$$

Quadratic upper bound for f

For $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$, we have, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$

$$f(\mathbf{x}) \leq f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 := Q_L(\mathbf{x}, \mathbf{y})$$

Quadratic *majorizer* for $f + g$ [24]

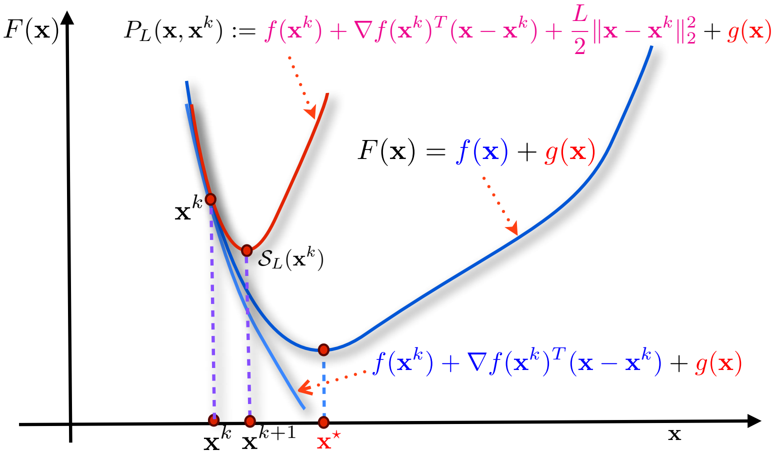
Of course, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$,

$$f(\mathbf{x}) \leq Q_L(\mathbf{x}, \mathbf{y}) \quad \Rightarrow \quad f(\mathbf{x}) + g(\mathbf{x}) \leq Q_L(\mathbf{x}, \mathbf{y}) + g(\mathbf{x}) := P_L(\mathbf{x}, \mathbf{y})$$

Proximal-gradient from the majorize-minimize perspective [24]

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} P_L(\mathbf{x}, \mathbf{x}^k) = \mathrm{prox}_{g/L}(\mathbf{x} - \nabla f(\mathbf{x}^k)/L)$$

Geometric illustration



Proximal-gradient algorithm

Basic proximal-gradient scheme (ISTA) [13, 25]
1. Choose $\mathbf{x}^0 \in \mathrm{dom}(F)$ arbitrarily as a starting point. **2.** For $k = 0, 1, \cdots$, generate a sequence $\{\mathbf{x}^k\}_{k \geq 0}$ as: $$\mathbf{x}^{k+1} := \mathrm{prox}_{\alpha g}\left(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)\right),$$ where $\alpha := \frac{1}{L}$.

Proximal-gradient algorithm

Basic proximal-gradient scheme (ISTA) **[13, 25]**
1. Choose $\mathbf{x}^0 \in \mathrm{dom}(F)$ arbitrarily as a starting point.   **2.** For $k = 0, 1, \cdots$, generate a sequence $\{\mathbf{x}^k\}_{k\geq 0}$ as: $$\mathbf{x}^{k+1} := \mathrm{prox}_{\alpha g}\left(\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k)\right),$$ where $\alpha := \frac{1}{L}$.

Theorem (Convergence of ISTA [4])

Let $\{\mathbf{x}^k\}$ be generated by ISTA. Then:

$$F(\mathbf{x}^k) - F^\star \leq \frac{L_f\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2}{2(k+1)}$$

The worst-case complexity to reach $F(\mathbf{x}^k) - F^\star \leq \varepsilon$ of (ISTA) *is $\mathcal{O}\left(\frac{L_f R_0^2}{\varepsilon}\right)$, where $R_0 := \max_{\mathbf{x}^\star\in\mathcal{S}^\star} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2$.*

- A line-search procedure can be used to estimate L_k for L based on $(0 < c \leq 1)$:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) - \frac{c}{2L_k}\|\nabla f(\mathbf{x}^k)\|^2.$$

A non-exhaustive list of proximal tractability functions

Name	Function	Proximal operator	Complexity
ℓ_1-norm	$f(\mathbf{x}) := \Vert\mathbf{x}\Vert_1$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = \mathrm{sign}(\mathbf{x}) \otimes [\vert\mathbf{x}\vert - \alpha]_+$	$\mathcal{O}(p)$
ℓ_2-norm	$f(\mathbf{x}) := \Vert\mathbf{x}\Vert_2$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = [1 - \alpha/\Vert\mathbf{x}\Vert_2]_+\mathbf{x}$	$\mathcal{O}(p)$
Support function	$f(\mathbf{x}) := \max_{\mathbf{y}\in\mathcal{C}} \mathbf{x}^T\mathbf{y}$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = \mathbf{x} - \alpha\pi_{\mathcal{C}}(\mathbf{x})$	
Box indicator	$f(\mathbf{x}) := \iota_{[\mathbf{a},\mathbf{b}]}(\mathbf{x})$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = \pi_{[\mathbf{a},\mathbf{b}]}(\mathbf{x})$	$\mathcal{O}(p)$
Positive semidefinite cone indicator	$f(\mathbf{X}) := \iota_{\mathbb{S}^p_+}(\mathbf{X})$	$\mathrm{prox}_{\alpha f}(\mathbf{X}) = \mathbf{U}[\Sigma]_+\mathbf{U}^T$, where $\mathbf{X} = \mathbf{U}\Sigma\mathbf{U}^T$	$\mathcal{O}(p^3)$
Hyperplane indicator	$f(\mathbf{x}) := \iota_{\mathcal{X}}(\mathbf{x})$, $\mathcal{X} := \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = b\}$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = \pi_{\mathcal{X}}(\mathbf{x}) = \mathbf{x} + \left(\frac{b-\mathbf{a}^T\mathbf{x}}{\Vert\mathbf{a}\Vert_2}\right)\mathbf{a}$	$\mathcal{O}(p)$
Simplex indicator	$f(\mathbf{x}) = \iota_{\mathcal{X}}(\mathbf{x}), \mathcal{X} := \{\mathbf{x} : \mathbf{x} \geq 0,\ \mathbf{1}^T\mathbf{x} = 1\}$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = (\mathbf{x} - \nu\mathbf{1})$ for some $\nu \in \mathbb{R}$, which can be efficiently calculated	$\tilde{\mathcal{O}}(p)$
Convex quadratic	$f(\mathbf{x}) := (1/2)\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x}$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = (\alpha\mathbb{I} + \mathbf{Q})^{-1}\mathbf{x}$	$\mathcal{O}(p\log p) \rightarrow \mathcal{O}(p^3)$
Square ℓ_2-norm	$f(\mathbf{x}) := (1/2)\Vert\mathbf{x}\Vert_2^2$	$\mathrm{prox}_{\alpha f}(\mathbf{x}) = (1/(1+\alpha))\mathbf{x}$	$\mathcal{O}(p)$
log-function	$f(\mathbf{x}) := -\log(x)$	$\mathrm{prox}_{\alpha f}(x) = ((x^2 + 4\alpha)^{1/2} + x)/2$	$\mathcal{O}(1)$
$\log\det$-function	$f(\mathbf{x}) := -\log\det(\mathbf{X})$	$\mathrm{prox}_{\alpha f}(\mathbf{X})$ is the log-function prox applied to the individual eigenvalues of $\mathbf{X}$	$\mathcal{O}(p^3)$

Here: $[\mathbf{x}]_+ := \max\{0, \mathbf{x}\}$ and $\iota_{\mathcal{X}}$ is the indicator function of the convex set $\mathcal{X}$, sign is the sign function, $\mathbb{S}^p_+$ is the cone of symmetric positive semidefinite matrices.

For more functions, see [12, 54].

Fast proximal-gradient algorithm

Fast proximal-gradient scheme (FISTA)
1. Choose $\mathbf{x}^0 \in \mathrm{dom}(F)$ arbitrarily as a starting point.
2. Set $\mathbf{y}^0 := \mathbf{x}^0$ and $t_0 := 1$.
3. For $k = 0, 1, \ldots$, generate two sequences $\{\mathbf{x}^k\}_{k \geq 0}$ and $\{\mathbf{y}^k\}_{k \geq 0}$ as:
$$\begin{cases} \mathbf{x}^{k+1} & := \mathrm{prox}_{\alpha g}\left(\mathbf{y}^k - \alpha \nabla f(\mathbf{y}^k)\right), \\ t_{k+1} & := (1 + \sqrt{4t_k^2 + 1})/2, \\ \mathbf{y}^{k+1} & := \mathbf{x}^{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k). \end{cases}$$
where $\alpha := L^{-1}$.

From $\mathcal{O}\left(\frac{L_f R_0^2}{\epsilon}\right)$ to $\mathcal{O}\left(R_0 \sqrt{\frac{L_f}{\epsilon}}\right)$ iterations at almost no additional cost!.

Fast proximal-gradient algorithm

Fast proximal-gradient scheme (FISTA)
1. Choose $\mathbf{x}^0 \in \text{dom}(F)$ arbitrarily as a starting point. **2.** Set $\mathbf{y}^0 := \mathbf{x}^0$ and $t_0 := 1$. **3.** For $k = 0, 1, \ldots$, generate two sequences $\{\mathbf{x}^k\}_{k\geq 0}$ and $\{\mathbf{y}^k\}_{k\geq 0}$ as: $$\begin{cases} \mathbf{x}^{k+1} & := \text{prox}_{\alpha g}\left(\mathbf{y}^k - \alpha\nabla f(\mathbf{y}^k)\right), \\ t_{k+1} & := (1 + \sqrt{4t_k^2 + 1})/2, \\ \mathbf{y}^{k+1} & := \mathbf{x}^{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k). \end{cases}$$ where $\alpha := L^{-1}$.

From $\mathcal{O}\left(\frac{L_f R_0^2}{\epsilon}\right)$ to $\mathcal{O}\left(R_0\sqrt{\frac{L_f}{\epsilon}}\right)$ iterations at almost no additional cost!.

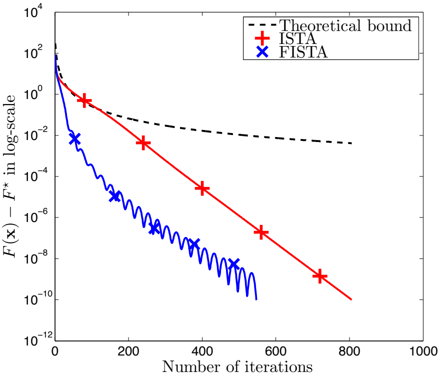
Complexity per iteration

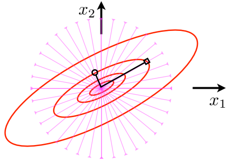
- **One** gradient $\nabla f(\mathbf{y}^k)$ and **one** prox-operator of g;
- 8 arithmetic operations for t_{k+1} and γ_{k+1};
- 2 more vector additions, and **one** scalar-vector multiplication.

The **cost per iteration** is almost the same as in **gradient scheme** if proximal operator of g is efficient.

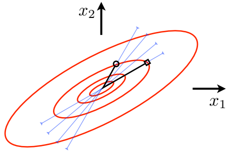
Example 1: Theoretical bounds vs practical performance

- **Theoretical bound: FISTA** $:= \frac{2L_f R_0^2}{(k+2)^2}$.





descent directions



restricted descent directions

- ℓ_1-regularized least squares formulation has **restricted strong convexity**. The proximal-gradient method can automatically exploit this structure.

Example 2: Sparse logistic regression

Problem (Sparse logistic regression [34])

Given $\mathbf{A} \in \mathbb{R}^{n\times p}$ *and* $\mathbf{b} \in \{-1,+1\}^n$, *solve:*

$$F^\star := \min_{\mathbf{x},\beta} \left\{ F(\mathbf{x}) := \frac{1}{n}\sum_{j=1}^{n} \log\left(1 + \exp\left(-\mathbf{b}_j(\mathbf{a}_j^T\mathbf{x} + \beta)\right)\right) + \rho\|\mathbf{x}\|_1 \right\}.$$

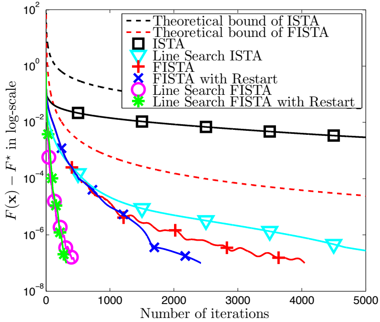
Real data

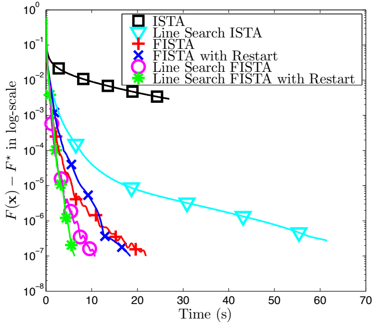
- Real data: `w8a` with $n = 49'749$ data points, $p = 300$ features
- Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

Parameters

- $\rho = 10^{-4}$.
- Number of iterations 5000, tolerance 10^{-7}.
- Ground truth: Solve problem up to 10^{-9} accuracy by TFOCS to get a high accuracy approximation of $\mathbf{x}^\star$ and $F^\star$.

Example 2: Sparse logistic regression - numerical results





	ISTA	LS-ISTA	FISTA	FISTA-R	LS-FISTA	LS-FISTA-R
Number of iterations	5000	5000	4046	2423	447	317
CPU time (s)	26.975	61.506	21.859	18.444	10.683	6.228
Solution error $(\times 10^{-7})$	29370	2.774	1.000	0.998	0.961	0.985

Summary of the worst-case complexities

Comparison with gradient scheme ($F(\mathbf{x}^k) - F^\star \leq \varepsilon$)

Complexity	Proximal-gradient scheme	Fast proximal-gradient scheme
Complexity [$\mu = 0$]	$\mathcal{O}\left(R_0^2(L_f/\varepsilon)\right)$	$\mathcal{O}\left(R_0\sqrt{L_f/\varepsilon}\right)$
Per iteration	1-gradient, 1-prox, 1-sv, 1-$v+$	1-gradient, 1-prox, 2-sv, 3-$v+$
Complexity [$\mu > 0$]	$\mathcal{O}\left(\kappa\log(\varepsilon^{-1})\right)$	$\mathcal{O}\left(\sqrt{\kappa}\log(\varepsilon^{-1})\right)$
Per iteration	1-gradient, 1-prox, 1-sv, 1-$v+$	1-gradient, 1-prox, 1-sv, 2-$v+$

Here: sv = scalar-vector multiplication, $v+$=vector addition.
$R_0 := \max_{\mathbf{x}^\star \in \mathcal{S}^\star} \|\mathbf{x}^0 - \mathbf{x}^\star\|$ and $\kappa = L_f/\mu_f$ is the condition number.

Need alternatives when

- computing $\nabla f(\mathbf{x})$ is much costlier than computing prox_g
- f is self-concordant

Software

TFOCS is a good software package to learn about first order methods.

`http://cvxr.com/tfocs/`

Examples

Example (Sparse graphical model selection)

$$\min_{\Theta \succ 0} \Big\{ \underbrace{\mathrm{tr}(\Sigma\Theta) - \log\det(\Theta)}_{f(\mathbf{x})} + \underbrace{\rho\|\mathrm{vec}(\Theta)\|_1}_{g(\mathbf{x})} \Big\}$$

where $\Theta \succ 0$ means that Θ is symmetric and positive definite, and $\rho > 0$ is a regularization parameter and vec is the vectorization operator.

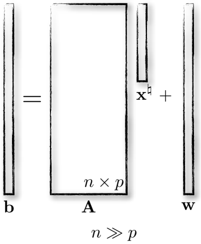
- Computing the gradient is expensive: $\nabla f(\Theta) = \Theta^{-1}$.
- $f \in \mathcal{F}_2$ is self-concordant. However, if $\alpha\mathbf{I} \preceq \Theta \preceq \beta\mathbf{I}$, then $f \in \mathcal{F}^{2,1}_{L,\mu}$ with $L = \sqrt{p}/\alpha^2$ and $\mu = (\beta^2\sqrt{p})^{-1}$.

Example (ℓ_1-regularized Lasso)

$$\min_{\mathbf{x}} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2}_{f(\mathbf{x})} + \underbrace{\rho\|\mathbf{x}\|_1}_{g(\mathbf{x})}$$

where $n \gg p$, $\mathbf{A} \in \mathbb{R}^{n\times p}$ is a full column-rank matrix, and $\rho > 0$ is a regularization parameter.

- $f \in \mathcal{F}^{2,1}_{L,\mu}$ and computing the gradient is $\mathcal{O}(n)$.



Variable metric proximal-gradient algorithm

Variable metric proximal-gradient algorithm [61]
1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$. **2**. For $k = 0, 1, \cdots$, perform: $$\begin{cases} \mathbf{d}^k & := \mathrm{prox}_{\mathbf{H}_k g}\left(\mathbf{x}^k - \mathbf{H}_k \nabla f(\mathbf{x}^k)\right) - \mathbf{x}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$ where $\alpha_k \in (0, 1]$ is a given step size. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

$\mathbf{H_k}$ incorporates both a step-size and a preconditioner.

Variable metric proximal operator

Given $\mathbf{H} \succ 0$ and $g \in \mathcal{F}(\mathbb{R}^p)$. The variable metric proximal operator of g is defined as

$$\mathrm{prox}_{\mathbf{H} g}(\mathbf{x}) := \arg\min_{\mathbf{y} \in \mathbb{R}^p} \left\{ g(\mathbf{y}) + (1/2)(\mathbf{y} - \mathbf{x})^T \mathbf{H}^{-1} (\mathbf{y} - \mathbf{x}) \right\}$$

Variable metric proximal-gradient algorithm

Variable metric proximal-gradient algorithm [61]

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := \mathrm{prox}_{\mathbf{H}_k g}\left(\mathbf{x}^k - \mathbf{H}_k \nabla f(\mathbf{x}^k)\right) - \mathbf{x}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

$\mathbf{H_k}$ incorporates both a step-size and a preconditioner.

Common choices of $\mathbf{H}_k$

- $\mathbf{H}_k := \lambda_k \mathbb{I}$, we have $\mathrm{prox}_{\mathbf{H}g} \equiv \mathrm{prox}_{\lambda g}$ and obtain a proximal-gradient method.
- $\mathbf{H}_k := \mathbf{D}$ (a positive diagonal matrix), $\mathrm{prox}_{\mathbf{H}g}$ can be transformed into $\mathrm{prox}_{\lambda g}$ (by scaling the variables) and we obtain a proximal-gradient method.
- $\mathbf{H}_k := \nabla^2 f(\mathbf{x}^k)^{-1}$, we obtain a proximal-Newton method.
- $\mathbf{H}_k \approx \nabla^2 f(\mathbf{x}^k)^{-1}$, we obtain a proximal quasi-Newton method.

Proximal-Newton method for composite self-concordant min.

Proximal-Newton algorithm (PNA)[4]
1. Choose $\mathbf{x}^0 \in \mathrm{dom}(F)$ as a starting point. **2**. For $k = 0, 1, \cdots,$ perform: $$\begin{cases} \mathbf{B}_k & := \nabla^2 f(\mathbf{x}^k) \quad ({\mathbf{H}_k}^{-1} = \mathbf{B}_k) \\ \mathbf{d}^k & := \mathrm{prox}_{\mathbf{B}_k^{-1} g}\left(\mathbf{x}^k - \mathbf{B}_k^{-1}\nabla f(\mathbf{x}^k)\right) - \mathbf{x}^k, \quad \text{(PN direction)} \\ \lambda_k & := \lVert \mathbf{d} \rVert_{\mathbf{x}^k}, \quad \text{(PN decrement)} \\ \alpha_k & = (1 + \lambda_k)^{-1}, \quad \text{(step-size)} \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k. \end{cases}$$

Complexity-per-iteration

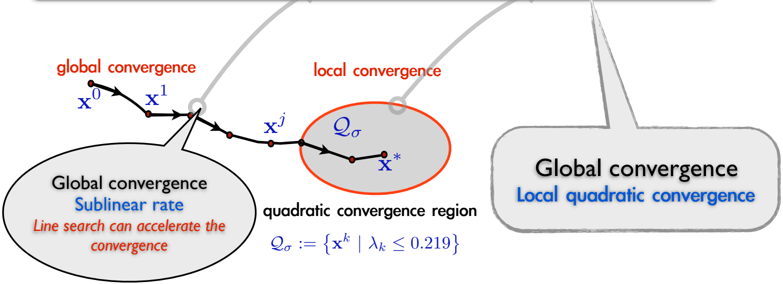
- Evaluation of $\nabla^2 f(\mathbf{x}^k)$ and $\nabla f(\mathbf{x}^k)$ (closed form expressions).
- Computing $\mathrm{prox}_{\mathbf{H}_k g}$ requires to solve a strongly convex program.
- Computing proximal-Newton decrement λ_k requires $(\mathbf{d}^k)^T \nabla f^2(\mathbf{x}^k)\mathbf{d}^k$.

[4] Recall: f is M_f-self concordant if $|\varphi'''(t)| \leq M_f \varphi''(t)^{3/2}$ with $\varphi(t) := f(\mathbf{x} + t\mathbf{v})$.

Overall analytical worst-case complexity

The worst-case analytical complexity: $F(\mathbf{x}^k) - F^\star \leq \varepsilon$

$$\#\text{iterations}(k) = \left\lfloor \frac{F(\mathbf{x}^0) - F^\star}{0.021} \right\rfloor + O\left(\ln\ln\left(\frac{4.56}{\varepsilon}\right)\right)$$



Example: Graphical model selection

Graphical model selection

$$\min_{\Theta \succ 0} \left\{ \underbrace{\mathrm{tr}(\Sigma\Theta) - \log\det(\Theta)}_{f(\Theta)} + \underbrace{\rho\|\mathrm{vec}(\Theta)\|_1}_{g(\Theta)} \right\}.$$

Example: Graphical model selection

Graphical model selection

$$\min_{\Theta \succ 0} \left\{ \underbrace{\mathrm{tr}(\Sigma\Theta) - \log\det(\Theta)}_{f(\Theta)} + \underbrace{\rho\|\mathrm{vec}(\Theta)\|_1}_{g(\Theta)} \right\}.$$

Computational cost

- $\nabla f(\Theta) = \mathrm{vec}(\Sigma - \Theta_k^{-1})$ and $\nabla^2 f(\Theta^k) = \Theta_k^{-1} \otimes \Theta_k^{-1}$ ($\otimes$-Kronecker product).
- Compute the **search direction** $\mathbf{d}_k$.

$$\mathbf{U}_k = \operatorname*{argmin}_{\|\mathrm{vec}(\mathbf{U})\|_1 \leq 1} \left\{ (1/2)\mathrm{trace}((\Theta_k \mathbf{U})^2) + \mathrm{trace}(\mathbf{Q}_k \mathbf{U}) \right\},$$

where $\mathbf{Q}_k := \rho^{-1}(\Theta_k \Sigma \Theta_k - 2\Theta_k)$. Then $\mathbf{d}^k := -((\Theta_k \Sigma - \mathbb{I})\Theta_k + \rho\Theta_k \mathbf{U}_k \Theta_k)$.
- The proximal-Newton decrement λ_k:

$$\lambda_k := (p - 2\mathrm{trace}(\mathbf{W}_k) + \mathrm{trace}(\mathbf{W}_k^2))^{1/2}, \quad \mathbf{W}_k := \Theta_k(\Sigma + \rho\mathbf{U}_k).$$

Example: Graphical model selection

Graphical model selection

$$\min_{\Theta \succ 0} \left\{ \underbrace{\mathrm{tr}(\Sigma\Theta) - \log\det(\Theta)}_{f(\Theta)} + \underbrace{\rho\|\mathrm{vec}(\Theta)\|_1}_{g(\Theta)} \right\}.$$

Computational cost

- $\nabla f(\Theta) = \mathrm{vec}(\Sigma - \Theta_k^{-1})$ and $\nabla^2 f(\Theta^k) = \Theta_k^{-1} \otimes \Theta_k^{-1}$ ($\otimes$-Kronecker product).
- Compute the **search direction** $\mathbf{d}_k$ via dualization:

$$\mathbf{U}_k = \operatorname*{argmin}_{\|\mathrm{vec}(\mathbf{U})\|_1 \le 1} \left\{ (1/2)\mathrm{trace}((\Theta_k \mathbf{U})^2) + \mathrm{trace}(\mathbf{Q}_k \mathbf{U}) \right\},$$

  where $\mathbf{Q}_k := \rho^{-1}(\Theta_k \Sigma \Theta_k - 2\Theta_k)$. Then $\mathbf{d}^k := -((\Theta_k\Sigma - \mathbb{I})\Theta_k + \rho\Theta_k\mathbf{U}_k\Theta_k)$.
- The proximal-Newton decrement λ_k:

$$\lambda_k := (p - 2\mathrm{trace}(\mathbf{W}_k) + \mathrm{trace}(\mathbf{W}_k^2))^{1/2}, \quad \mathbf{W}_k := \Theta_k(\Sigma + \rho\mathbf{U}_k).$$

Only need **matrix-matrix multiplications**

No Cholesky factorizations or matrix inversions

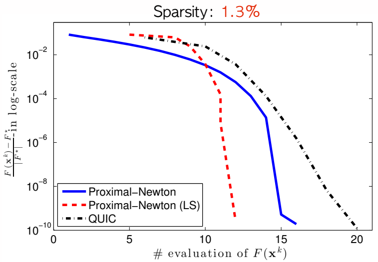
cf. Lecture 5 @ `http://lions.epfl.ch/mathematics_of_data`

Test on the real-data: Lymph and Leukemia

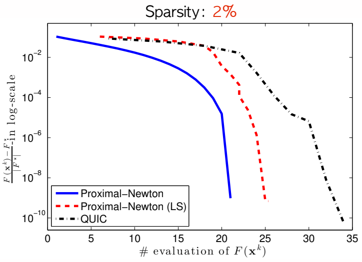
- **PNA vs. QUIC:**
  - QUIC subproblem solver: special block-coordinate descent algorithm.
  - PNA subproblem solver: general proximal-gradient algorithms.

On the average $\times 5$ acceleration (up to $\times 15$) over Matlab QUIC

- **Convergence behavior:** $\rho = 0.5$ - Gene data (Genetic regulatory network)



Lymph [p = 587] ~ 350,000 variables



Leukemia [p = 1255] ~ 1.5 millions variables

[4] Details: Composite self-concordant minimization, Journal of Machine Learning Research, vol. 16, 2015

Swiss army knife of convex formulations

Our **primal problem** prototype: A simple mathematical formulation[5]

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \in \mathcal{X} \Big\},$$

- $f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex function, and $\mathcal{X}$ is a nonempty, closed convex set.
- $\mathbf{A} \in \mathbb{R}^{n\times p}$ and $\mathbf{b} \in \mathbb{R}^n$ are known.
- An optimal solution $\mathbf{x}^\star$ satisfies $f(\mathbf{x}^\star) = f^\star$, $\mathbf{A}\mathbf{x}^\star = \mathbf{b}$ and $\mathbf{x}^\star \in \mathcal{X}$.

[5] We can simply replace $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{C}$ for a convex cone $\mathcal{C}$ without fundamental changes.

Swiss army knife of convex formulations

Our **primal problem** prototype: A simple mathematical formulation[5]

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \in \mathcal{X} \Big\},$$

- $f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex function, and $\mathcal{X}$ is a nonempty, closed convex set.
- $\mathbf{A} \in \mathbb{R}^{n\times p}$ and $\mathbf{b} \in \mathbb{R}^n$ are known.
- An optimal solution $\mathbf{x}^\star$ satisfies $f(\mathbf{x}^\star) = f^\star$, $\mathbf{A}\mathbf{x}^\star = \mathbf{b}$ and $\mathbf{x}^\star \in \mathcal{X}$.

Example to keep in mind in the sequel

$$\mathbf{x}^\star := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ \|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}, \|\mathbf{x}\|_\infty \leq 1 \Big\}$$

[5] We can simply replace $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{C}$ for a convex cone $\mathcal{C}$ without fundamental changes.

Swiss army knife of convex formulations

Our **primal problem** prototype: A simple mathematical formulation[5]

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \in \mathcal{X} \Big\},$$

- $f : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex function, and $\mathcal{X}$ is a nonempty, closed convex set.
- $\mathbf{A} \in \mathbb{R}^{n\times p}$ and $\mathbf{b} \in \mathbb{R}^n$ are known.
- An optimal solution $\mathbf{x}^\star$ satisfies $f(\mathbf{x}^\star) = f^\star$, $\mathbf{A}\mathbf{x}^\star = \mathbf{b}$ and $\mathbf{x}^\star \in \mathcal{X}$.

Example to keep in mind in the sequel

$$\mathbf{x}^\star := \arg\min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ \|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}, \|\mathbf{x}\|_\infty \leq 1 \Big\}$$

Broader context

- Standard convex optimization formulations: *linear programming, convex quadratic programming, second order cone programming, semidefinite programming, and geometric programming.*
- Reformulations of existing unconstrained problems via **convex splitting**: *composite convex minimization, and consensus optimization, ...*

[5] We can simply replace $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{C}$ for a convex cone $\mathcal{C}$ without fundamental changes.

Numerical ϵ-accuracy

Exact vs. approximate solutions

- Computing an **exact solution** $\mathbf{x}^\star$ is **impracticable** unless problem has a closed form solution, which is extremely limited in reality.
- Numerical optimization algorithms result in $\mathbf{x}^\star_\epsilon$ that approximates $\mathbf{x}^\star$ up to a given accuracy ϵ in **some sense**.

- In the sequel, by ϵ-accurate solutions $\mathbf{x}^\star_\epsilon$, we mean the following

[6] Very often, $\mathcal{X}$ is a "simple set." Hence, requiring $\mathbf{x}^\star_\epsilon \in \mathcal{X}$ is acceptable in practice.*

*** I will absorb $\mathcal{X}$ into the objective f with a so-called indicator function in the next slide to ease the notation.**

Numerical ϵ-accuracy

Exact vs. approximate solutions

- Computing an **exact solution** $\mathbf{x}^\star$ is **impracticable** unless problem has a closed form solution, which is extremely limited in reality.
- Numerical optimization algorithms result in $\mathbf{x}^\star_\epsilon$ that approximates $\mathbf{x}^\star$ up to a given accuracy ϵ in **some sense**.

- In the sequel, by ϵ-accurate solutions $\mathbf{x}^\star_\epsilon$, we mean the following

Definition (ϵ-accurate solutions)

Given a numerical tolerance $\epsilon \geq 0$, a point $\mathbf{x}^\star_\epsilon \in \mathbb{R}^p$ is called an ϵ-solution if

$$\begin{cases} |f(\mathbf{x}^\star_\epsilon) - f^\star| \leq \epsilon & \text{(objective residual)},\\ \|\mathbf{A}\mathbf{x}^\star_\epsilon - \mathbf{b}\| \leq \epsilon & \text{(feasibility gap)},\\ \mathbf{x}^\star_\epsilon \in \mathcal{X} & \text{(exact simple set feasibility)}.^6 \end{cases}$$

- When $\mathbf{x}^\star$ is unique, we can also obtain $\|\mathbf{x}^\star_\epsilon - \mathbf{x}^\star\| \leq \epsilon$ (iterate residual).
- Indeed, ϵ can be different for the objective, feasibility gap, or the iterate residual.

[6] Very often, $\mathcal{X}$ is a "simple set." Hence, requiring $\mathbf{x}^\star_\epsilon \in \mathcal{X}$ is acceptable in practice.*

* **I will absorb $\mathcal{X}$ into the objective f with a so-called indicator function in the next slide to ease the notation.**

The optimal solution set

Before we talk about algorithms, we must first characterize what we are looking for!

Lagrange function and the minimax formulation

We can naturally interpret the optimality condition via a minimax formulation

$$\max_{\lambda} \min_{\mathbf{x} \in \mathrm{dom}(f)} \mathcal{L}(\mathbf{x}, \lambda),$$

where $\lambda \in \mathbb{R}^n$ is the vector of Lagrange multipliers or dual variables w.r.t. $\mathbf{A}\mathbf{x} = \mathbf{b}$ associated with the Lagrange function:

$$\mathcal{L}(\mathbf{x}, \lambda) := f(\mathbf{x}) + \lambda^T(\mathbf{A}\mathbf{x} - \mathbf{b}).$$

The optimal solution set

Before we talk about algorithms, we must first characterize what we are looking for!

Lagrange function and the minimax formulation

We can naturally interpret the optimality condition via a minimax formulation

$$\max_{\lambda} \min_{\mathbf{x} \in \mathrm{dom}(f)} \mathcal{L}(\mathbf{x}, \lambda),$$

where $\lambda \in \mathbb{R}^n$ is the vector of Lagrange multipliers or dual variables w.r.t. $\mathbf{A}\mathbf{x} = \mathbf{b}$ associated with the Lagrange function:

$$\mathcal{L}(\mathbf{x}, \lambda) := f(\mathbf{x}) + \lambda^T(\mathbf{A}\mathbf{x} - \mathbf{b}).$$

Optimality condition

The **optimality condition** of $\min_{\mathbf{x} \in \mathbb{R}^p} \{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ can be written as

$$\begin{cases} 0 & \in \mathbf{A}^T\lambda^\star + \partial f(\mathbf{x}^\star), \\ 0 & = \mathbf{A}\mathbf{x}^\star - \mathbf{b}. \end{cases}$$

(Subdifferential) $\partial f(\mathbf{x}) := \{\mathbf{v} \in \mathbb{R}^p \ : \ f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{v}^T(\mathbf{y} - \mathbf{x}), \ \forall \mathbf{y} \in \mathbb{R}^p\}$.

- This is the well-known KKT (Karush-Kuhn-Tucker) condition.
- Any point $(\mathbf{x}^\star, \lambda^\star)$ satisfying the optimality condition is called a KKT point.
- $\mathbf{x}^\star$ is called a stationary point and $\lambda^\star$ is the corresponding multipliers.

Finding an optimal solution

A plausible strategy

To solve the constrained problem: $\min_{\mathbf{x}}\{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, we therefore seek the solutions

$$(\mathbf{x}^\star, \lambda^\star) \in \arg\max_{\lambda} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda),$$

which we can naively break down into two—in general **nonsmooth**—problems:

Lagrangian subproblem:

$$\mathbf{x}^*(\lambda) \in \arg\min_{\mathbf{x}}\{\mathcal{L}(\mathbf{x}, \lambda) := f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x} - \mathbf{b}\rangle\}.$$

Dual problem:

$$\lambda^\star \in \arg\max_{\lambda} \{d(\lambda) := \mathcal{L}(\mathbf{x}^*(\lambda), \lambda)\}.$$

- $d(\cdot)$ is called the dual function, and the optimal dual value is $d^\star = d(\lambda^\star)$.

Since $d(\cdot)$ is concave, we can attempt the following strategy:

1. Find the optimal solution $\lambda^\star$ of the "convex" dual problem.
2. Obtain the optimal primal solution $\mathbf{x}^\star = \mathbf{x}^*(\lambda^\star)$ via the Lagrangian subproblem.

Finding an optimal solution

A plausible strategy

To solve the constrained problem: $\min_{\mathbf{x}}\{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, we therefore seek the solutions

$$(\mathbf{x}^\star, \lambda^\star) \in \arg\max_{\lambda} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda),$$

which we can naively break down into two—in general **nonsmooth**—problems:

Lagrangian subproblem:

$$\mathbf{x}^*(\lambda) \in \arg\min_{\mathbf{x}}\{\mathcal{L}(\mathbf{x}, \lambda) := f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x} - \mathbf{b}\rangle\}.$$

Dual problem:

$$\lambda^\star \in \arg\max_{\lambda} \left\{d(\lambda) := \mathcal{L}(\mathbf{x}^*(\lambda), \lambda)\right\}.$$

- $d(\cdot)$ is called the dual function, and the optimal dual value is $d^\star = d(\lambda^\star)$.

Since $d(\cdot)$ is concave, we can attempt the following strategy:

1. Find the optimal solution $\lambda^\star$ of the "convex" dual problem.
2. Obtain the optimal primal solution $\mathbf{x}^\star = \mathbf{x}^*(\lambda^\star)$ via the Lagrangian subproblem.

Challenges for the plausible strategy above

1. Establishing its correctness
2. Computational efficiency of finding an $\bar{\epsilon}$-approximate optimal dual solution $\lambda^\star_{\bar{\epsilon}}$
3. Mapping $\lambda^\star_{\bar{\epsilon}} \to \mathbf{x}^\star_{\epsilon}$ (i.e., $\bar{\epsilon}(\epsilon)$), where ϵ is for the original constrained problem

Finding an optimal solution

A plausible strategy

To solve the constrained problem: $\min_{\mathbf{x}}\{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, we therefore seek the solutions

$$(\mathbf{x}^\star, \lambda^\star) \in \arg\max_{\lambda} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda),$$

which we can naively break down into two—in general **nonsmooth**—problems:

Lagrangian subproblem:

$$\mathbf{x}^*(\lambda) \in \arg\min_{\mathbf{x}}\{\mathcal{L}(\mathbf{x}, \lambda) := f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x} - \mathbf{b}\rangle\}.$$

Dual problem:

$$\lambda^\star \in \arg\max_{\lambda} \{d(\lambda) := \mathcal{L}(\mathbf{x}^*(\lambda), \lambda)\}.$$

- $d(\cdot)$ is called the dual function, and the optimal dual value is $d^\star = d(\lambda^\star)$.

Since $d(\cdot)$ is concave, we can attempt the following strategy:

1. Find the optimal solution $\lambda^\star$ of the "convex" dual problem.
2. Obtain the optimal primal solution $\mathbf{x}^\star = \mathbf{x}^*(\lambda^\star)$ via the Lagrangian subproblem.

Challenges for the plausible strategy above

1. Establishing its correctness: Assume $f^\star > -\infty$ and Slater's condition for $f^\star = d^\star$
2. Computational efficiency of finding an $\bar{\epsilon}$-approximate optimal dual solution $\lambda^\star_{\bar{\epsilon}}$
3. Mapping $\lambda^\star_{\bar{\epsilon}} \to \mathbf{x}^\star_{\epsilon}$ (i.e., $\bar{\epsilon}(\epsilon)$), where ϵ is for the original constrained problem

Efficiency considerations for the dual problem

Dual subgradient method
1. Choose $\lambda^0 \in \mathbb{R}^n$.
2. For $k = 0, 1, \cdots$, perform: $\lambda^{k+1} = \lambda^k + \alpha_k \mathbf{v}^k,$
where $\mathbf{v}^k \in \partial d(\lambda^k)$ and α_k is the step-size.

Subgradient method for the dual

Assumptions:

1. $\|\mathbf{v}\|_2 \leq G$ for all $\mathbf{v} \in \partial d(\lambda)$, $\lambda \in \mathbb{R}^n$.
2. $\|\lambda^0 - \lambda^\star\|_2 \leq R$
3. Step-size: $\alpha_k = \frac{R}{G\sqrt{k}}$.

Conclusion:

$$\min_{0\leq i\leq k} d^\star - d(\lambda^i) \leq \frac{RG}{\sqrt{k}}$$

Efficiency considerations for the dual problem

Dual subgradient method
1. Choose $\lambda^0 \in \mathbb{R}^n$.
2. For $k = 0, 1, \cdots$, perform: $\lambda^{k+1} = \lambda^k + \alpha_k \mathbf{v}^k$,
where $\mathbf{v}^k \in \partial d(\lambda^k)$ and α_k is the step-size.

Subgradient method for the dual

Assumptions:

1. $\|\mathbf{v}\|_2 \leq G$ for all $\mathbf{v} \in \partial d(\lambda)$, $\lambda \in \mathbb{R}^n$.
2. $\|\lambda^0 - \lambda^\star\|_2 \leq R$
3. Step-size: $\alpha_k = \frac{R}{G\sqrt{k}}$.

Conclusion:

$$\min_{0 \leq i \leq k} d^\star - d(\lambda^i) \leq \frac{RG}{\sqrt{k}} \leq \bar{\epsilon}.$$

SGM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}^2}\right) \times$ subgradient calculation

Efficiency considerations for the dual problem

Dual gradient method
1. Choose $\lambda^0 \in \mathbb{R}^n$.
2. For $k = 0, 1, \cdots$, perform: $\lambda^{k+1} = \lambda^k + \frac{1}{L}\nabla d(\lambda^k)$,
where L is the Lipschitz constant.

Subgradient method for the dual

Assumptions:

1. $\|\mathbf{v}\|_2 \leq G$ for all $\mathbf{v} \in \partial d(\lambda)$, $\lambda \in \mathbb{R}^n$.
2. $\|\lambda^0 - \lambda^\star\|_2 \leq R$
3. Step-size: $\alpha_k = \frac{R}{G\sqrt{k}}$.

Conclusion:

$$\min_{0 \leq i \leq k} d^\star - d(\lambda^i) \leq \frac{RG}{\sqrt{k}} \leq \bar{\epsilon}.$$

SGM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}^2}\right) \times$ subgradient calculation

GM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}}\right) \times$ gradient calculation

Impact of Lipschitz gradient

- d is differentiable concave and has Lipschitz continuous gradient if:

$$\|\nabla d(\lambda) - \nabla d(\eta)\|_2 \leq L\|\lambda - \eta\|_2, \ \forall \eta, \lambda.$$

- We denote: $d \in \mathcal{F}_L$.
- If $d \in \mathcal{F}_L$, then the gradient method with step-size $1/L$ obeys:

$$d^\star - d(\lambda^k) \leq \frac{2LR^2}{k+4} \leq \bar{\epsilon}.$$

Efficiency considerations for the dual problem

Dual gradient method
1. Choose $\lambda^0 \in \mathbb{R}^n$.   **2**. For $k = 0, 1, \cdots$, perform: $\lambda^{k+1} = \lambda^k + \frac{1}{L}\nabla d(\lambda^k)$,   where L is the Lipschitz constant.

Subgradient method for the dual

Assumptions:

1. $\|\mathbf{v}\|_2 \leq G$ for all $\mathbf{v} \in \partial d(\lambda)$, $\lambda \in \mathbb{R}^n$.
2. $\|\lambda^0 - \lambda^\star\|_2 \leq R$
3. Step-size: $\alpha_k = \frac{R}{G\sqrt{k}}$.

Conclusion:

$$\min_{0 \leq i \leq k} d^\star - d(\lambda^i) \leq \frac{RG}{\sqrt{k}} \leq \bar{\epsilon}.$$

SGM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}^2}\right) \times$ subgradient calculation

GM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}}\right) \times$ gradient calculation

Impact of Lipschitz gradient

- d is differentiable concave and has Lipschitz continuous gradient if:

$$\|\nabla d(\lambda) - \nabla d(\eta)\|_2 \leq L\|\lambda - \eta\|_2, \ \forall \eta, \lambda.$$

- We denote: $d \in \mathcal{F}_L$.
- If $d \in \mathcal{F}_L$, then the gradient method with step-size $1/L$ obeys:

$$d^\star - d(\lambda^k) \leq \frac{2LR^2}{k+4} \leq \bar{\epsilon}.$$

This is NOT the best we can do.

- There exists a complexity lower-bound:

$$d^\star - d(\lambda^k) \geq \frac{3LR^2}{32(k+1)^2}, \forall d \in \mathcal{F}_L,$$

for any iterative method based only on function and gradient evaluations ($p \gg 1$).

Efficiency considerations for the dual problem

Dual accelerated gradient method

1. Choose $\hat{\lambda}^0 = \lambda^0 \in \mathbb{R}^n$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \lambda^{k+1} = \hat{\lambda}^k + \frac{1}{L}\nabla d(\hat{\lambda}^k), \\ \hat{\lambda}^{k+1} = \lambda^{k+1} + \rho_k(\lambda^{k+1} - \lambda^k), \end{cases}$$

where ρ_k is a momentum parameter.

Subgradient method for the dual

Assumptions:

1. $\|\mathbf{v}\|_2 \leq G$ for all $\mathbf{v} \in \partial d(\lambda)$, $\lambda \in \mathbb{R}^n$.
2. $\|\lambda^0 - \lambda^\star\|_2 \leq R$
3. Step-size: $\alpha_k = \frac{R}{G\sqrt{k}}$.

Conclusion:

$$\min_{0 \leq i \leq k} d^\star - d(\lambda^i) \leq \frac{RG}{\sqrt{k}} \leq \bar{\epsilon}.$$

SGM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}^2}\right) \times$ subgradient calculation

GM: $\mathcal{O}\left(\frac{1}{\bar{\epsilon}}\right) \times$ gradient calculation

AGM: $\mathcal{O}\left(\frac{1}{\sqrt{\bar{\epsilon}}}\right) \times$ gradient calculation

Impact of Lipschitz gradient

- d is differentiable concave and has Lipschitz continuous gradient if:

$$\|\nabla d(\lambda) - \nabla d(\eta)\|_2 \leq L\|\lambda - \eta\|_2, \;\forall \eta, \lambda.$$

- We denote: $d \in \mathcal{F}_L$.
- For all $d \in \mathcal{F}_L$, the accelerated gradient method with $\rho_k = \frac{k+1}{k+3}$ obeys:

$$d^\star - d(\lambda^k) \leq \frac{2LR^2}{(k+2)^2} \leq \bar{\epsilon}$$

This is NEARLY the best we can do.

- There exists a complexity lower-bound:

$$d^\star - d(\lambda^k) \geq \frac{3LR^2}{32(k+1)^2}, \forall d \in \mathcal{F}_L,$$

for any iterative method based only on function and gradient evaluations ($p \gg 1$).

We can use an **averaging scheme** to recover a primal solution [42, 67].

Primal-dual algorithms via model-based gap reduction technique

To characterize the primal and dual optimality of $\min_{\mathbf{x}\in\mathcal{X}} \{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, we define:

- The feasible set: $\mathcal{D} := \{\mathbf{x} \in \mathcal{X} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$.
- The primal-dual gap function:

$$G(\mathbf{z}) := f(\mathbf{x}) - d(\lambda)$$

  where $\mathbf{z} := (\mathbf{x}, \lambda)$.

Properties of the primal dual gap function

G defined has following properties:

1. $G(\mathbf{z}) \geq 0$ for all $\mathbf{x} \in \mathcal{D}$ and $\lambda \in \mathbb{R}^n$.
2. $G(\mathbf{z}^\star) = 0$ iff $\mathbf{z}^\star := (\mathbf{x}^\star, \lambda^\star)$ is the primal and dual optimal solutions.
3. G is **convex** but generally **nonsmooth**.

cf. Lectures 7 and 8 @ `http://lions.epfl.ch/mathematics_of_data`

Smoothing techniques

Smoothing functions

Let $b_{\mathcal{C}} : \mathcal{C} \subseteq \mathbb{R}^p \to \mathbb{R}$ be continuous and μ-strongly convex (with $\mu = 1$). We call $b_{\mathcal{C}}$ the prox-function.

Examples:

- $b(\mathbf{x}) := \frac{1}{2}\|\mathbf{x}\|_2^2$ is a prox-function of $\mathbb{R}^p$.
- $b(\mathbf{x}) := \sum_{i=1}^p \mathbf{x}_i \log(\mathbf{x}_i) + p$ is a prox-function of $\Delta_p := \left\{\mathbf{x} \in \mathbb{R}^p_+, \mathbf{1}^T\mathbf{x} = 1\right\}$.

Since G is nonsmooth, our idea is to smooth G using smoothing functions.

Smoothing the primal-dual gap function

Given two smoothness parameters $\gamma > 0$ and $\beta > 0$, we define an approximation of G:

$$G_{\gamma\beta}(\mathbf{z}) := f_\beta(\mathbf{x}) - d_\gamma(\lambda), \quad \text{where}$$

$$\begin{aligned} f_\beta(\mathbf{x}) &:= \max_{\lambda \in \mathbb{R}^n} \left\{ f(\mathbf{x}) + \langle \mathbf{A}\mathbf{x} - \mathbf{b}, \lambda \rangle - \frac{\beta}{2}\|\lambda\|_2^2 \right\} \equiv f(\mathbf{x}) + \frac{1}{2\beta}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \approx f(\mathbf{x}) \\ d_\gamma(\lambda) &:= \min_{\mathbf{x} \in \mathcal{X}} \left\{ f(\mathbf{x}) + \langle \mathbf{A}^T\lambda, \mathbf{x} \rangle + \gamma b(\mathbf{A}\mathbf{x}) \right\} \approx d(\lambda). \end{aligned}$$

Result: d_γ is Lipschitz continuous. Hence, $G_{\gamma\beta}$ is composite when f is proximal.

The primal-dual steps

In order to evaluate $G_{\gamma\beta}$, we need to solve:

$$\begin{cases} \mathbf{x}^*_\gamma(\lambda) & := \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) + (\mathbf{A}^T\lambda)^T\mathbf{x} + \gamma b(\mathbf{A}\mathbf{x}) \right\} \quad \text{(primal step)} \\ \lambda^*_\beta(\mathbf{x}) & := \beta^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b}) \quad \text{(dual step)}. \end{cases}$$

- Primal step: Requires the solution of the convex subproblem.
- Dual step: Requires **one** matrix-vector multiplication.

Decomposable structure of f and $\mathcal{X}$: Parallel computation

- Decomposable structure

$$f(\mathbf{x}) := \sum_{i=1}^m f_i(\mathbf{x}_i) \quad \text{and} \quad \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m.$$

- Choose $b(\mathbf{A}\mathbf{x}) := \sum_{i=1}^m b_i(\mathbf{A}_i\mathbf{x}_i)$, then $\mathbf{x}^*_\gamma(\lambda) := [\mathbf{x}^*_{\gamma,1}(\lambda), \cdots, \mathbf{x}^*_{\gamma,m}(\lambda)]$:

$$\mathbf{x}^*_{\gamma,i}(\lambda) := \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \left\{ f_i(\mathbf{x}_i) + (\mathbf{A}_i^T\lambda)^T\mathbf{x}_i + \gamma b_i(\mathbf{A}_i\mathbf{x}_i) \right\}.$$

- Choose $b(\mathbf{A}\mathbf{x}) := \frac{1}{2}\sum_{i=1}^m \|\mathbf{x}_i - \mathbf{x}_i^c\|_2^2$, then $\mathbf{x}^*_\gamma(\lambda) := [\mathbf{x}^*_{\gamma,1}(\lambda), \cdots, \mathbf{x}^*_{\gamma,m}(\lambda)]$:

$$\mathbf{x}^*_{\gamma,i}(\lambda) := \mathrm{prox}_{\gamma^{-1}f_i + \iota_{\mathcal{X}_i}}\left(\mathbf{x}_i^c - \gamma^{-1}\mathbf{A}_i^T\lambda\right),$$

where $\iota_{\mathcal{X}_i}$ is the indicator function of $\mathcal{X}_i$.

A special case

The augmented Lagrangian (AL) smoothing

We can choose $b(\mathbf{A}\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. The primal step becomes an AL step:

$$\mathbf{x}^*(\lambda) := \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) + \langle \mathbf{A}^T\lambda, \mathbf{x}\rangle + \frac{\gamma}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \right\}.$$

- $\mathbf{x}^*(\lambda)$ can be computed approximately by first-order methods.
- A warm-start reduces the iterations of such first-order algorithms.
- Large γ leads to less number of iterations but increases the difficulty of computing $\mathbf{x}^*(\lambda)$.

Key estimates

Model-based gap reduction condition

A **sequence** $\{\bar{\mathbf{z}}^k\}_{k\geq 0} \subset \mathcal{X} \times \mathbb{R}^n$ satisfies the model-based gap reduction condition if:

$$G_{\gamma_{k+1}\beta_{k+1}}(\bar{\mathbf{z}}^{k+1}) \leq (1-\tau_k)G_{\gamma_k\beta_k}(\bar{\mathbf{z}}^k) + \psi_k,$$

where $\psi_k \leq 0$ or ($\psi_k \geq 0$ and $\psi_k \to 0^+$), $\tau_k \in (0,1)$ and $\gamma_k\beta_{k+1} < \gamma_k\beta_k$ for $k \geq 0$.

Theorem (Bounds on the objective residual and primal feasibility)

We can generate a sequence $\{\bar{\mathbf{z}}^k\}$ in $\mathcal{X} \times \mathbb{R}^n$ with $\bar{\mathbf{z}}^k := (\bar{\mathbf{x}}^k, \bar{\lambda}^k)$ such that:

$$\begin{cases} |f(\bar{\mathbf{x}}^k) - f^\star| \leq \mathcal{O}\left(\gamma_k\right) \text{ or } \mathcal{O}\left(\beta_k\right), \\ \|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\| \leq \mathcal{O}(\beta_k). \end{cases}$$

Uncertainty principle

The parameters $(\gamma_k, \beta_k, \tau_k)$ are updated such that:

$$\gamma_k\beta_k = \Omega(\tau_k^2).$$

For the augmented Lagrangian smoother, we have $\gamma_k = \gamma > 0$, and $\beta_k = \mathcal{O}(\tau_k^2)$.

- Optimal rate for $\{\tau_k\}$: $\tau_k^2 = \Omega\left(\frac{1}{k^2}\right)$.

Accelerated gradient method (expanded)

The standard scheme ([49])

The accelerated scheme for minimizing $g \in \mathcal{F}_L^{1,1}$ consists of three main steps:

$$\begin{cases} \hat{\lambda}^k & := (1-\tau_k)\lambda^k + \tau_k \lambda_k^* \\ \lambda^{k+1} & := \hat{\lambda}^k - \frac{1}{L_g}\nabla g(\hat{\lambda}^k) \\ \lambda_{k+1}^* & := \lambda_k^* - \frac{1}{\tau_k}(\hat{\lambda}^k - \lambda^{k+1}). \end{cases}$$

Here, L_g is the Lipschitz constant of ∇g and $\tau_k \in (0,1)$ is a given momentum term.

Accelerated gradient scheme for the smoothed dual problem

Recall the smoothed dual function d_γ with $-d_\gamma \in \mathcal{F}_L^{1,1}$. The AGM for this problem can be written as

$$\begin{cases} \hat{\lambda}^k & := (1-\tau_k)\lambda^k + \tau_k \lambda_k^* \\ \lambda^{k+1} & := \hat{\lambda}^k + \frac{\gamma}{L_d}(\mathbf{A}\mathbf{x}_\gamma^*(\hat{\lambda}^k) - \mathbf{b}) \\ \lambda_{k+1}^* & := \lambda_k^* - \frac{1}{\tau_k}(\hat{\lambda}^k - \lambda^{k+1}). \end{cases} \tag{1}$$

Here, $L_d > 0$, (e.g., $L_d := \|\mathbf{A}\|^2$ or $L_d := 1$) and $\nabla d_\gamma(\hat{\lambda}^k) = \mathbf{A}\mathbf{x}_\gamma^*(\hat{\lambda}^k) - \mathbf{b}$.

Our primal-dual scheme

The primal-dual scheme (http://lions.epfl.ch/decopt)

Our approach is fundamentally the same as the accelerated gradient method:

$$\begin{cases} \hat{\lambda}^k & := (1-\tau_k)\lambda^k + \tau_k \lambda_k^* & \text{(dual acceleration step)} \\ \lambda^{k+1} & := \hat{\lambda}^k + \frac{\gamma_{k+1}}{L_d}(\mathbf{A}\mathbf{x}^*_{\gamma_{k+1}}(\hat{\lambda}^k) - \mathbf{b}) & \text{(primal acceleration step)} \\ \bar{\mathbf{x}}^{k+1} & := (1-\tau_k)\bar{\mathbf{x}}^k + \tau_k \mathbf{x}^*_{\gamma_{k+1}}(\hat{\lambda}^k) & \\ \lambda^*_{k+1} & := \frac{1}{\beta_{k+1}}(\mathbf{A}\bar{\mathbf{x}}^{k+1} - \mathbf{b}) & \text{(dual gradient update).} \end{cases} \tag{2}$$

Both smoothing parameters γ and β are updated at each iteration.

The correspondance between (1) and (2)

The last step of (2) (vs. (1)) is split into two steps:

$$\frac{1}{\beta_{k+1}}(\mathbf{A}\bar{\mathbf{x}}^{k+1} - \mathbf{b}) = \frac{1}{\beta_k}(\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}) + \frac{\gamma_{k+1}}{\tau_k L_d}(\mathbf{A}\mathbf{x}^*_{\gamma_{k+1}}(\hat{\lambda}^k) - \mathbf{b}).$$

Using $\bar{\mathbf{x}}^{k+1} := (1-\tau_k)\bar{\mathbf{x}}^k + \tau_k \mathbf{x}^*_{\gamma_{k+1}}(\hat{\lambda}^k)$ we can show that:

$$\begin{cases} \beta_{k+1} = (1-\tau_k)\beta_k \\ \beta_{k+1}\gamma_{k+1} = \tau_k^2 L_d. \end{cases}$$

Convergence guarantee and an extension

Theorem [60, 59]

1. When f is strongly convex with $\mu > 0$, we can take $\gamma_k = 0$ and $\beta_k = \mathcal{O}\left(\frac{1}{k^2}\right)$:

$$\begin{cases} -D_{\Lambda^\star}\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\| \leq & f(\mathbf{x}^k) - f^\star & \leq 0 \\ & \|\mathbf{A}\mathbf{x}^k - \mathbf{b}\| & \leq \frac{4\|\mathbf{A}\|^2}{(k+2)^2\mu} D_{\Lambda^\star} \\ & \|\mathbf{x}^k - \mathbf{x}^\star\| & \leq \frac{4\|\mathbf{A}\|}{(k+2)\mu} D_{\Lambda^\star} \end{cases}$$

2. When f is non-smooth, the best we can do is $\gamma_k = \mathcal{O}\left(\frac{1}{k}\right)$ and $\beta_k = \mathcal{O}\left(\frac{1}{k}\right)$:

$$\begin{cases} -D_{\Lambda^\star}\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\| \leq & f(\mathbf{x}^k) - f^\star & \leq \frac{C_p D_{\mathcal{X}}}{k+1}, \\ & \|\mathbf{A}\mathbf{x}^k - \mathbf{b}\| & \leq \frac{C_d(D_{\Lambda^\star} + \sqrt{D_{\mathcal{X}}})}{k+1}, \end{cases}$$

where C_p and C_d are two given positive constants depending on different schemes.

Handling a cone constraint $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$

$$\begin{cases} \lambda^{k+1} & := \mathrm{proj}_{\mathcal{K}^*}\left(\hat{\lambda}^k + \frac{\gamma}{\|\mathbf{A}\|^2}(\mathbf{A}\mathbf{x}^*_\gamma(\hat{\lambda}^k) - \mathbf{b})\right) \\ \lambda^*_{k+1} & := \arg\max_{\lambda \in \mathcal{K}^*} \left\{\langle \mathbf{A}\bar{\mathbf{x}}^{k+1} - \mathbf{b}, \lambda\rangle - \beta_{k+1} b(\lambda)\right\}. \end{cases}$$

Here, $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$, $\mathrm{proj}_{\mathcal{K}^*}$ is the projection onto $\mathcal{K}^*$, and b is a chosen proximity function.

Example: An application of the convergence guarantees

Problem (Consensus optimization)

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p}\left\{f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{x})\right\}$$

Constrained reformulation via a product space trick with $\bar{\mathbf{z}}^k := [\bar{\mathbf{x}}_1^k, \ldots, \bar{\mathbf{x}}_n^k]$*:*

$$F^\star := \min_{\mathbf{z}:=[\mathbf{x}_1,\ldots,\mathbf{x}_n]\in\mathbb{R}^{np}}\left\{F(\mathbf{z}) := \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{x}_i) : \mathbf{x}_i - \mathbf{x}_j = 0, (i,j)\in E\right\}$$

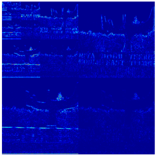
for some user-defined graph $\mathcal{G} = (V, E)$ *with vertices* V *and edges* E*.*

Interpretation of the convergence guarantees

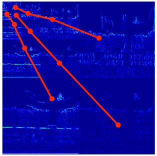
By using our algorithm in a decentralized but synchronized fashion, we obtain

$$|F(\bar{\mathbf{z}}^k) - f^\star| \leq \mathcal{O}(1/k) \quad \text{and} \quad \sum_{(i,j)\in E}\|\bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_j^k\|^2 \leq \mathcal{O}(1/k^2),\ i = 1, \ldots, n-1.$$

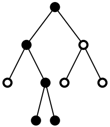
Tree sparsity [33, 18, 2, 69]



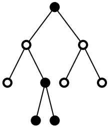
Wavelet coefficients



Wavelet tree



Valid selection of nodes



Invalid selection of nodes

Structure: *We seek the sparsest signal with a rooted connected subtree support.*

Optimization formulation (TU-relax [21])

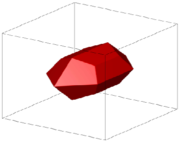
$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^p} \quad & f(\mathbf{x}) := \textstyle\sum_{\mathcal{G}_i\in\mathfrak{G}} \|\mathbf{x}_{\mathcal{G}_i}\|_\infty \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned}$$

This problem possesses two key structures: decomposability and tractable proximity.

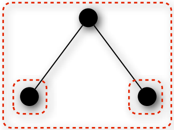
When $g = p$ and $\mathcal{G}_i = \{i\}$, this problem reduces to the well-known basis pursuit (BP):

$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b}.$$

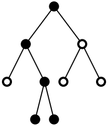
Tree sparsity [33, 18, 2, 69]



$f(\mathbf{x})$-ball



$\mathfrak{G} = \{\{1,2,3\},\{2\},\{3\}\}$



valid selection of nodes

Structure: *We seek the sparsest signal with a rooted connected subtree support.*

Optimization formulation (TU-relax [21])

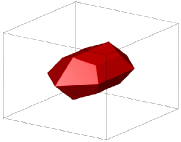
$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & f(\mathbf{x}) := \sum_{\mathcal{G}_i\in\mathfrak{G}} \|\mathbf{x}_{\mathcal{G}_i}\|_\infty \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{array}$$

This problem possesses two key structures: decomposability and tractable proximity.

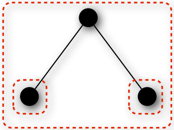
When $g = p$ and $\mathcal{G}_i = \{i\}$, this problem reduces to the well-known basis pursuit (BP):

$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b}.$$

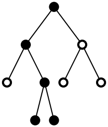
Tree sparsity [33, 18, 2, 69]



$f(\mathbf{x})$-ball



$\mathfrak{G} = \{\{1,2,3\},\{2\},\{3\}\}$



valid selection of nodes

Structure: *We seek the sparsest signal with a rooted connected subtree support.*

Optimization formulation (TU-relax [21])

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & f(\mathbf{x}) := \sum_{\mathcal{G}_i\in\mathfrak{G}} \|\mathbf{x}_{\mathcal{G}_i}\|_\infty + \rho\|\boldsymbol{\Psi}\mathbf{x}\|_{\mathsf{TV}} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{array}$$

This problem possesses two key structures: decomposability and tractable proximity.

When $g = p$ and $\mathcal{G}_i = \{i\}$, this problem reduces to the well-known basis pursuit (BP):

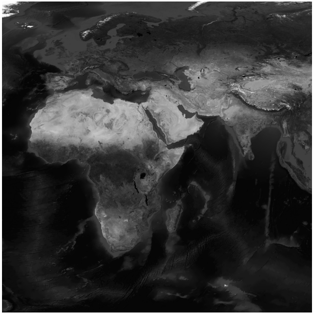
$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{x}\|_1 + \rho\|\boldsymbol{\Psi}\mathbf{x}\|_{\mathsf{TV}} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b}.$$

Adding additional regularizers to BP does not pose any difficulty ($\boldsymbol{\Psi}$ is the wavelet transform and ρ is a regularization parameter).

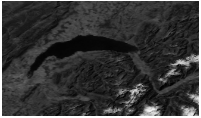
Tree sparsity example: 1:100-compressive sensing [59, 60, 1]

Problem dimensions: $(p = 10^9,\ n = 10^7)$

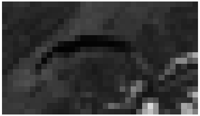
World [1Gpix]



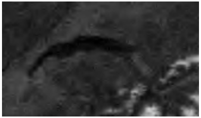
Lac Léman



World [10Mpix]

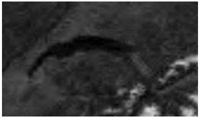


sparse



PNSR $= 31.83$db

tree-sparse



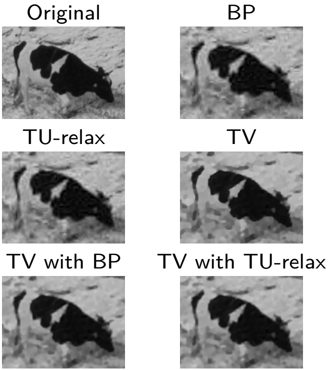
PNSR $= 32.48$db

Augmented Lagrangian smoothing:

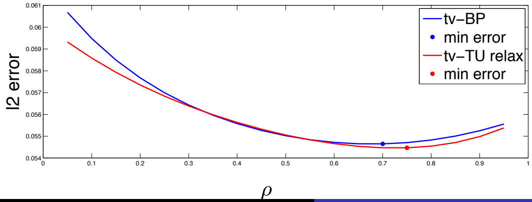
- Iterations: 113
- Primal-dual gap: $1\mathrm{e}{-8}$
- Number of $(\mathbf{A}, \mathbf{A}^T)$ applications: $(684, 570)$
- Time: < 4 days.

Tree sparsity example: TV & TU-relax 1:15-compression [62, 1]

Original tiff image $[2048 \times 2048]$



Regularization:



Primal-dual methods: The zoo

- **Plenty ...** many of them are relatively popular, which are not covered here.

cf. Lectures 7 and 8 @ `http://lions.epfl.ch/mathematics_of_data`

- **Primal-dual methods** are rooted in **Arrow-Hurwitz's method**, when applied to the composite convex problem: $\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) + g(\mathbf{K}\mathbf{x})$, which is equivalent to:

$$\min_{\mathbf{x}} \max_{\mathbf{z}} \left\{ f(\mathbf{x}) + \langle \mathbf{K}\mathbf{x}, \mathbf{z} \rangle - g^*(\mathbf{z}) \right\}.$$

  - Chambolle-Pock's algorithm [8], and its variants, e.g., He-Yuan's variant [30].
  - Primal-dual Hybrid Gradient (PDHG) method and its variants [22, 28].
  - Proximal-based decomposition (Chen-Teboulle's algorithm) [10].

- **Primal dual methods** are rooted in splitting techniques from monotone inclusions:
  - Primal-dual splitting algorithms [3, 6, 11, 64, 14, 15].
  - Three-operator splitting [16], parallel variants, ...

Primal-dual methods: The zoo (cont.)

- **Primal dual methods** are rooted in splitting techniques when applied to the dual:
  - Alternating minimization algorithms (AMA) [26, 64].
  - Alternating direction methods of multipliers (ADMM) [20, 31].
  - Accelerated variants of AMA and ADMM [15, 29]
  - Preconditioned ADMM, Linearized ADMM and inexact Uzawa algorithms [8, 52].
- **Primal-dual second order decomposition methods:**
  - Dual (quasi) Newton methods [66].
  - Smoothing decomposition methods via barriers functions [41, 63, 68].
  - Look forward to our new ADMM/AMA methods (coming up soon!)

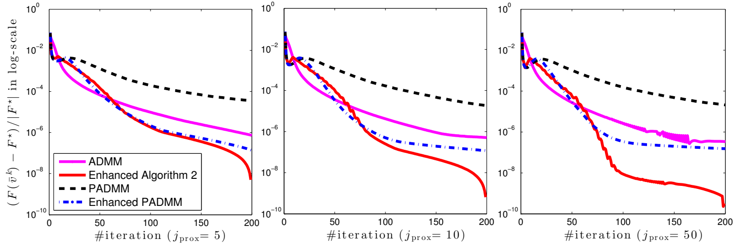
Example: Poison imaging reconstruction

Poisson imaging reconstruction based on patches

$$\min_{\mathbf{x}\in\mathbb{R}^{n\times p}} \Big\{ \underbrace{\sum_{i=1}^{n}(\mathbf{K}\mathbf{x})_i - \sum_{i=1}^{n} b_i \log((\mathbf{K}\mathbf{x})_i)}_{f(\mathbf{x})} + \underbrace{\rho\|\mathbf{x}\|_{\mathcal{S}}}_{g(\mathbf{x})} \Big\}.$$

- $\mathbf{K}$ is a blur operator, $\mathbf{b} \in \mathbb{Z}^n_+$ is the observed vector of photon counts.
- $\rho > 0$ is a regularization parameter, and $\|\mathbf{x}\|_{\mathrm{S}} = \|\mathcal{H}(\mathrm{mat}(\mathbf{x}))\|_*$ is the nuclear norm of $\mathcal{H}(\mathrm{mat}(\mathbf{x}))$, where $\mathcal{H}$ constructs a patch based mapping, see [37].

Example [62]: An application to the confocal microscopy problem



Note that $\mathbb{I} + \mathbf{K}^T\mathbf{K}$ is Fourier diagonalizable, which makes ADDM efficient [23].

Revisiting the prox-operator

Prox-operator helps us process nonsmooth terms “efficiently”

$$\mathrm{prox}_g(\mathbf{x}) := \arg\min_{\mathbf{z}\in\mathbb{R}^p}\{g(\mathbf{z}) + (1/2)\|\mathbf{z}-\mathbf{x}\|^2\}.$$

Key properties:

- single valued & non-expansive.
- distributes when the primal problem has decomposable structure:

$$f(\mathbf{x}) := \sum_{i=1}^{m} f_i(\mathbf{x}_i), \quad \text{and} \quad \mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_m.$$

  where $m \geq 1$ is the number of components.
- often efficient & has closed form expression. For instance, if $g(\mathbf{z}) = \|\mathbf{z}\|_1$, then the prox-operator performs coordinate-wise soft-thresholding by 1.

Revisiting the prox-operator

Prox-operator helps us process nonsmooth terms "efficiently"

$$\mathrm{prox}_g(\mathbf{x}) := \arg\min_{\mathbf{z}\in\mathbb{R}^p}\{g(\mathbf{z}) + (1/2)\|\mathbf{z}-\mathbf{x}\|^2\}.$$

Key properties:

- single valued & non-expansive.
- distributes when the primal problem has decomposable structure:

$$f(\mathbf{x}) := \sum_{i=1}^{m} f_i(\mathbf{x}_i), \quad \text{and} \quad \mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_m.$$

  where $m \geq 1$ is the number of components.
- often efficient & has closed form expression. For instance, if $g(\mathbf{z}) = \|\mathbf{z}\|_1$, then the prox-operator performs coordinate-wise soft-thresholding by 1.

Not all nonsmooth functions are proximal-friendly!

If $g(\mathbf{z}) = \|\mathbf{z}\|_\star$ (i.e., the nuclear norm of $\mathbf{z}$) then the prox-operator may require a full singular value decomposition.

We can sometimes avoid the prox-operator!

Example: Frank-Wolfe's method

Problem setting

$$f^\star := \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

Assumptions

- $\mathcal{X}$ is nonempty, convex, closed and bounded.
- $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$ (i.e., convex with Lipschitz gradient).
- Note that $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$ is missing from our prototype problem

Frank-Wolfe's method (see [32] for a review)

Conditional gradient method (CGA)

1. Choose $\mathbf{x}^0 \in \mathcal{X}$.

2. For $k = 0, 1, \cdots,$ perform:

$$\begin{cases} \hat{\mathbf{x}}^k & := \arg\min_{\mathbf{x}\in\mathcal{X}} \nabla f(\mathbf{x}^k)^T\mathbf{x}, \\ \mathbf{x}^{k+1} & := (1-\gamma_k)\mathbf{x}^k + \gamma_k\hat{\mathbf{x}}^k, \end{cases}$$

where $\gamma_k := \frac{2}{k+2}$ is a given relaxation parameter.

Example: Frank-Wolfe's method

Problem setting

$$f^\star := \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

Assumptions

- $\mathcal{X}$ is nonempty, convex, closed and bounded.
- $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$ (i.e., convex with Lipschitz gradient).
- Note that $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$ is missing from our prototype problem

Frank-Wolfe's method (see [32] for a review)

Conditional gradient method (CGA)

1. Choose $\mathbf{x}^0 \in \mathcal{X}$.

2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \hat{\mathbf{x}}^k & := \arg\min_{\mathbf{x}\in\mathcal{X}} \nabla f(\mathbf{x}^k)^T\mathbf{x}, \quad (*) \\ \mathbf{x}^{k+1} & := (1-\gamma_k)\mathbf{x}^k + \gamma_k\hat{\mathbf{x}}^k, \end{cases}$$

where $\gamma_k := \frac{2}{k+2}$ is a given relaxation parameter.

When $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^{n\times p} : \|\mathbf{x}\|_\star \leq 1\}$, $(*)$ corresponds to rank-1 updates!

Towards Fenchel-type operators

Generalized sharp operators [67]

We define the (generalized) sharp operator of a convex function g over $\mathcal{X}$ as follows

$$[\mathbf{x}]^\sharp_{\mathcal{X},g} := \operatorname*{argmax}_{\mathbf{z}\in\mathcal{X}} \{\langle \mathbf{x}, \mathbf{z}\rangle - g(\mathbf{z})\}.$$

Important special cases:

1. If $g = 0$, then we obtain the so-called linear minimization oracle.
2. If $\mathcal{X} = \mathrm{dom}(g)$, then $[\mathbf{x}]^\sharp_g = \nabla g^*(\mathbf{x})$, where g^* is the Fenchel conjugate of g.

Example (Nuclear norm)

Consider $g(\mathbf{x}) := \frac{1}{2}\|\mathbf{x}\|_\star^2$ and $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^{n\times p} : \|\mathbf{x}\|_\star \leq 1\}$. Let $\mathbf{u}$ and $\mathbf{v}$ be the left and right principal singular vectors of $\mathbf{x}$ respectively. Then,

$$\mathbf{u}\mathbf{v}^T \in [\mathbf{x}]^\sharp_{\mathcal{X}} := [\mathbf{x}]^\sharp_{\mathcal{X},0}, \qquad \|\mathbf{x}\|\mathbf{u}\mathbf{v}^T \in [\mathbf{x}]^\sharp_g := [\mathbf{x}]^\sharp_{\mathbb{R}^{n\times p},g}$$

where $\|\cdot\|$ is the spectral norm. **The computations are essentially the same.**

Revisiting Frank-Wolfe's method

Problem setting

$$f^\star := \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

Assumptions

- $\mathcal{X}$ is nonempty, convex, closed and bounded.
- $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$ (i.e., convex with Lipschitz gradient).
- Note that $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$ is missing from our prototype problem

Frank-Wolfe's method (see [32] for a review)

Conditional gradient method (CGA)

1. Choose $\mathbf{x}^0 \in \mathcal{X}$.

2. For $k = 0, 1, \cdots,$ perform:

$$\begin{cases} \hat{\mathbf{x}}^k & := \arg\min_{\mathbf{x}\in\mathcal{X}} \nabla f(\mathbf{x}^k)^T\mathbf{x} \quad \equiv [\nabla f(\mathbf{x}^k)]^\sharp_{\mathcal{X}}, \\ \mathbf{x}^{k+1} & := (1-\gamma_k)\mathbf{x}^k + \gamma_k\hat{\mathbf{x}}^k, \end{cases}$$

where $\gamma_k := \frac{2}{k+2}$ is a given relaxation parameter.

Generalized conditional gradient method replaces the indicator function $\iota_{\mathcal{X}}$ with g:

$$\hat{\mathbf{x}}^k := \arg\min_{\mathbf{x}}\{g(\mathbf{x}) + \nabla f(\mathbf{x}^k)^T\mathbf{x}\} = [\nabla f(\mathbf{x}^k)]^\sharp_g.$$

Revisiting Frank-Wolfe's method

Problem setting

$$f^\star := \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

Assumptions

- $\mathcal{X}$ is nonempty, convex, closed and bounded.
- $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^p)$ (i.e., convex with Lipschitz gradient).
- **We will handle $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$ and nonsmooth $f(\mathbf{x})$ in the sequel!**

Frank-Wolfe's method (see [32] for a review)

Conditional gradient method (CGA)

1. Choose $\mathbf{x}^0 \in \mathcal{X}$.

2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \hat{\mathbf{x}}^k & := \arg\min_{\mathbf{x}\in\mathcal{X}} \nabla f(\mathbf{x}^k)^T\mathbf{x} \quad \equiv [\nabla f(\mathbf{x}^k)]^\sharp_{\mathcal{X}}, \\ \mathbf{x}^{k+1} & := (1-\gamma_k)\mathbf{x}^k + \gamma_k \hat{\mathbf{x}}^k, \end{cases}$$

where $\gamma_k := \frac{2}{k+2}$ is a given relaxation parameter.

Generalized conditional gradient method replaces the indicator function $\iota_{\mathcal{X}}$ with g:

$$\hat{\mathbf{x}}^k := \arg\min_{\mathbf{x}}\{g(\mathbf{x}) + \nabla f(\mathbf{x}^k)^T\mathbf{x}\} = [\nabla f(\mathbf{x}^k)]^\sharp_g.$$

Exploring the smoothness of the dual function in depth

Definition (Hölder continuous gradients [45])

Let us consider the following unconstrained setup

$$\min_{\mathbf{x}\in\mathbb{R}^p} g(\mathbf{x}).$$

A convex function g has Hölder continuous subgradients of degree $\nu \in [0,1]$ if there are two constants ν and $M_\nu \geq 0$ that satisfy:

$$\|\nabla g(\mathbf{x}) - \nabla g(\mathbf{y})\|_* \leq M_\nu \|\mathbf{x}-\mathbf{y}\|^\nu$$

where ∇g is a (sub)gradient of g.

Highlights:

1. $\nu = 1$ is the Lipschitz continuous gradients case where $L = M_\nu$.
2. $\nu = 0$ is the bounded gradient assumption (recall the subgradient method).
3. Iteration lowerbound for the Hölder class: $\mathcal{O}\left(\left(\frac{M_\nu\|\mathbf{x}^0-\mathbf{x}^\star\|^{1+\nu}}{\epsilon}\right)^{\frac{2}{1+3\nu}}\right)$.
4. The condition also ensures a basic surrogate

$$g(\mathbf{y}) \leq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y}-\mathbf{x}\rangle + \frac{M_\nu}{1+\nu}\|\mathbf{x}-\mathbf{y}\|^{1+\nu}$$

   for any $\mathbf{x},\mathbf{y} \in \mathcal{X}$.

Nesterov's universal gradient methods

Nesterov's universal gradient methods [48]

In practice, smoothness parameters ν and M_ν are usually not known. Nesterov's algorithms adapt to the unknown ν via an appropriate line-search strategy:

$$g(\mathbf{y}) \leq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{M}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \frac{\delta}{2}.$$

where inexactness parameter $\delta > 0$ depends only on the desired final accuracy.

They are universal since they ensure the best possible rate of convergence for each ν.

[7] PGM in [48] uses the Bregman / prox setup.

Nesterov's universal gradient methods

Nesterov's universal gradient methods [48]

In practice, smoothness parameters ν and M_ν are usually not known. Nesterov's algorithms adapt to the unknown ν via an appropriate line-search strategy:

$$g(\mathbf{y}) \leq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{M}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \frac{\delta}{2}.$$

where inexactness parameter $\delta > 0$ depends only on the desired final accuracy.

They are universal since they ensure the best possible rate of convergence for each ν.

Universal primal gradient method (PGM)[7]
1. Choose $\mathbf{x}^0 \in \mathcal{X}$, $M_{-1} > 0$ and accuracy $\epsilon > 0$.   **2.** For $k = 0, 1, \cdots$, perform: $$\mathbf{x}^{k+1} = \mathbf{x}^k - M_k^{-1}\nabla g(\mathbf{x}^k)$$ where we use line-search to find $M_k \geq 0.5M_{k-1}$ that satisfies: $$g(\mathbf{x}^{k+1}) \leq g(\mathbf{x}^k) + \langle \nabla g(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle + \frac{M_k}{2}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2 + \frac{\epsilon}{2}$$

[7] PGM in [48] uses the Bregman / prox setup.

Nesterov's universal gradient methods

Nesterov's universal gradient methods [48]

In practice, smoothness parameters ν and M_ν are usually not known. Nesterov's algorithms adapt to the unknown ν via an appropriate line-search strategy:

$$g(\mathbf{y}) \leq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{M}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \frac{\delta}{2}.$$

where inexactness parameter $\delta > 0$ depends only on the desired final accuracy.

They are universal since they ensure the best possible rate of convergence for each ν.

Universal primal gradient method (PGM)[7]
1. Choose $\mathbf{x}^0 \in \mathcal{X}$, $M_{-1} > 0$ and accuracy $\epsilon > 0$. **2.** For $k = 0, 1, \cdots$, perform: $\mathbf{x}^{k+1} = \mathbf{x}^k - M_k^{-1}\nabla g(\mathbf{x}^k)$ where we use line-search to find $M_k \geq 0.5M_{k-1}$ that satisfies: $g(\mathbf{x}^{k+1}) \leq g(\mathbf{x}^k) + \langle \nabla g(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle + \frac{M_k}{2}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2 + \frac{\epsilon}{2}$

Yes, there is an accelerated version.

[7] PGM in [48] uses the Bregman / prox setup.

Universal primal-dual decomposition methods

Our strategy: Hölder smoothness in the dual

Instead of smoothing, we assume that the dual function d is Hölder continuous for some $\nu \in [0, 1]$:

$$M_\nu(d) := \sup_{\lambda \neq \eta} \frac{\|\nabla d(\lambda) - \nabla d(\eta)\|_*}{\|\lambda - \eta\|^\nu}, \qquad M_d^* := \inf_{0 \leq \nu \leq 1} M_\nu(d) < +\infty.$$

We will solve the dual problem by a new FISTA version [4] of Nesterov's universal gradient algorithm [48] and develop new primal strategies to approximate $\mathbf{x}^\star$.

Is this assumption reasonable?

Consider two special cases:

- if $\mathcal{X}$ is bounded and d is subdifferentiable, then ∇d is also bounded.
- if f is uniformly convex with convexity parameter $\mu_f > 0$ and degree $q \geq 2$, i.e.,

$$\langle \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq \mu_f \|\mathbf{x} - \mathbf{y}\|^q$$

  for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, then ∇d satisfies Hölder condition with $\nu = \frac{1}{q-1}$ and $M_\nu = \left(\mu_f^{-1} \|\mathbf{A}\|^2\right)^{\frac{1}{q-1}}$.

Our universal primal-dual decomposition methods: The dual steps

Dual steps ([67])

- The universal dual gradient step:

$$\lambda^{k+1} := \lambda^k + \frac{1}{M_k}\nabla d(\lambda^k) = \lambda_k + \frac{1}{M_k}\left(\mathbf{A}\mathbf{x}^*(\lambda^k) - \mathbf{b}\right),$$

where $\mathbf{x}^*(\lambda^k)$ is computed via the sharp operator:

$$\mathbf{x}^*(\lambda^k) := \arg\min_{\mathbf{x}\in\mathbb{R}^p}\left\{f(\mathbf{x}) + \langle \mathbf{A}^T\lambda^k, \mathbf{x}\rangle\right\} \equiv \left[-\mathbf{A}^T\lambda^k\right]_f^\sharp.$$

- The universal dual accelerated gradient step:

$$\begin{cases} t_k & := 0.5\left(1 + \sqrt{1 + 4t_{k-1}^2}\right) \\ \hat{\lambda}^k & := \lambda^k + \frac{t_{k-1}-1}{t_k}\left(\lambda^k - \hat{\lambda}^{k-1}\right) \\ \lambda^{k+1} & := \hat{\lambda}^k + \frac{1}{M_k}\left(\mathbf{A}\mathbf{x}^*(\hat{\lambda}^k) - \mathbf{b}\right). \end{cases}$$

Line-search condition

The local smoothness constant M_k is computed via a line-search procedure:

$$d(\lambda^{k+1}) \geq d(\lambda^k) + \langle \nabla d(\lambda^k), \lambda^{k+1} - \lambda^k\rangle - \frac{M_k}{2}\|\lambda^{k+1} - \lambda^k\|^2 - \frac{\delta_k}{2}.$$

- $\delta_k = \epsilon$ for our universal dual gradient method
- $\delta_k = \epsilon/t_k$ for our universal dual accelerated gradient method

On the line-search

Number of line-search iterations

- Each line-search step costs one dual function evaluation.
- **(Acc)UniProx** requires (1)2 line-search steps per iteration on the average.
- In many cases, we can avoid the search step and find the step-size in one shot by solving an analytic equation obtained by using a proper bound on $d(\lambda^{k+1})$.

Example (Nuclear norm)

Consider $f := \frac{1}{2}\|\mathbf{x}\|_{\star}^2$ with the linear constraint $\mathbf{A}(\mathbf{x}) = \mathbf{b}$, which leads to the dual function $d(\lambda) = -\frac{1}{2}\|\mathbf{A}^T(\lambda)\|^2 - \langle \lambda, \mathbf{b}\rangle$. Using triangular inequality, we get

$$\begin{aligned} U(M_k) :&= d(\lambda^k) - \frac{\alpha_k^2}{2}\|\mathbf{A}^T(\nabla d(\lambda^k))\|^2 - \alpha_k\left[\|\mathbf{A}^T(\nabla d(\lambda^k))\|\|\mathbf{A}^T(\lambda^k)\| + \langle \nabla d(\lambda^k), \mathbf{b}\rangle\right] \\ &\leq d(\lambda^k + \frac{1}{M_k}\nabla d(\lambda^k)) = d(\lambda^{k+1}). \end{aligned}$$

We can solve the following second order equation

$$U(M_k) = d(\lambda^k) + \frac{\alpha_k}{2}\left\|\nabla d(\lambda^k)\right\|^2 - \frac{\delta_k}{2}$$

to find the step size $\alpha_k := \frac{1}{M_k}$ which guarantees the line-search condition.

The primal steps and the worst-case complexity

Primal steps - averaging steps

- The universal primal gradient step:

$$\textbf{(UniProx):} \qquad \bar{\mathbf{x}}^k := \left(\sum_{i=0}^{k} \frac{1}{M_i}\right)^{-1} \sum_{i=0}^{k} \frac{1}{M_i} \mathbf{x}^*(\lambda^i).$$

- The universal primal accelerated gradient step:

$$\textbf{(AccUniProx):} \qquad \bar{\mathbf{x}}^k := \left(\sum_{i=0}^{k} \frac{t_i}{M_i}\right)^{-1} \sum_{i=0}^{k} \frac{t_i}{M_i} \mathbf{x}^*(\lambda^i).$$

The worst-case complextity

To achieve $\bar{\mathbf{x}}^k$ such that $|f(\bar{\mathbf{x}}^k) - f^\star| \le \epsilon$ and $\|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\| \le \epsilon$ is:

$$\begin{cases} \text{For } \textbf{(UniProx):} & \mathcal{O}\left(D_{\Lambda^\star}^2 \inf_{0\le\nu\le1}\left(\frac{M_\nu}{\epsilon}\right)^{\frac{2}{1+\nu}}\right), \quad (\text{optimal for } \nu = 0). \\ \text{For } \textbf{(AccUniProx):} & \mathcal{O}\left(D_{\Lambda^\star}^{\frac{2+5\nu}{1+3\nu}} \inf_{0\le\nu\le1}\left(\frac{M_\nu}{\epsilon}\right)^{\frac{2}{1+3\nu}}\right), \quad (\text{optimal for } \nu = 1). \end{cases}$$

Summary of the algorithms and convergence guarantees - I

Universal primal-dual gradient method (UniProx)
Initialization: Choose $\lambda^0 \in \mathbb{R}^n$ and $\epsilon > 0$. Estimate a value $M_{-1} < 2M_\epsilon$.
Iteration: For $k = 0, 1, \cdots$, perform:
1. *Primal step:* $\mathbf{x}^*(\lambda^k) = [-\mathbf{A}^T\lambda^k]^\sharp_f$
2. *Dual gradient:* $\nabla d(\lambda^k) = \mathbf{A}^T\mathbf{x}^*(\lambda^k) - \mathbf{b}$
3. Line-search: Find $M_k \in [0.5M_{k-1}, 2M_\epsilon]$ from line-search condition and: $\lambda^{k+1} = \lambda^k + M_k^{-1}\nabla d(\lambda^k)$
4. *Primal averaging:* $\bar{\mathbf{x}}^k := S_k^{-1}\sum_{j=0}^k M_j^{-1}\mathbf{x}^*(\lambda^j)$ where $S_k = \sum_{j=0}^k M_j^{-1}$.

Theorem [67]

$\bar{\mathbf{x}}^k$ and $\bar{\lambda}^k := S_k^{-1}\sum_{j=0}^k M_j^{-1}\lambda^j$ obtained by **UniProx** satisfy (with $\lambda^0 = 0$):

$$
\begin{cases}
-\|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\|D_{\Lambda^\star} \leq & f(\bar{\mathbf{x}}^k) - f^\star & \leq \frac{\epsilon}{2}, \\
& \|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\| & \leq \frac{4M_\epsilon D_{\Lambda^\star}}{k+1} + \sqrt{\frac{2M_\epsilon \epsilon}{k+1}}, \\
& d^\star - d(\bar{\lambda}^k) & \leq \frac{M_\epsilon D^2_{\Lambda^\star}}{k+1} + \frac{\epsilon}{2}.
\end{cases}
$$

Summary of the algorithms and convergence guarantees - II

Accelerated universal primal-dual gradient method (AccUniProx)
Initialization: Choose $\lambda^0 \in \mathbb{R}^n$, $\epsilon > 0$. Set $t_0 = 1$. Estimate a value $M_{-1} < 2M_\epsilon$.
Iteration: For $k = 0, 1, \cdots$, perform:
1. *Primal step:* $\mathbf{x}^*(\hat{\lambda}^k) = [-\mathbf{A}^T\hat{\lambda}^k]^\sharp_f$,
2. *Dual gradient:* $\nabla d(\hat{\lambda}^k) = \mathbf{A}^T\mathbf{x}^*(\hat{\lambda}^k) - \mathbf{b}$,
3. *Line-search:* Find $M_k \in [M_{k-1}, 2M_\epsilon]$ from line-search condition and: $\lambda^{k+1} = \hat{\lambda}^k + M_k^{-1}\nabla d(\hat{\lambda}^k)$,
4. $t_{k+1} = 0.5[1 + \sqrt{1 + 4t_k^2}]$,
5. $\hat{\lambda}_{k+1} = \lambda_{k+1} + \frac{t_k - 1}{t_{k+1}}(\lambda_{k+1} - \lambda_k)$,
6. *Primal averaging:* $\bar{\mathbf{x}}^k := S_k^{-1}\sum_{j=0}^k t_j M_j^{-1}\mathbf{x}^*(\hat{\lambda}^j)$ where $S_k = \sum_{j=0}^k t_j M_j^{-1}$.

Theorem [67]

$\bar{\mathbf{x}}^k$ and λ^k obtained by **AccUniProx** satisfy (with $\lambda^0 = 0$):

$$
\begin{cases}
-\|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\| D_{\Lambda^\star} \leq & f(\bar{\mathbf{x}}^k) - f^\star \leq \frac{\epsilon}{2}, \\
& \|\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b}\| \leq \frac{16 M_\epsilon D_{\Lambda^\star}}{(k+2)^{\frac{1+3\nu}{1+\nu}}} + \sqrt{\frac{8M_\epsilon \epsilon}{k+2}}, \\
& d^\star - d(\lambda^k) \leq \frac{4 M_\epsilon D_{\Lambda^\star}^2}{(k+1)^{\frac{1+3\nu}{1+\nu}}} + \frac{\epsilon M_\epsilon}{M_0}(k+1)^{\frac{1-\nu}{1+\nu}}.
\end{cases}
$$

The dual may NOT converge for $(\nu = 0)$!

The general constraint case

Handling to the constraint $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}$

Dual steps need to be changed:

- The universal dual gradient step:

$$\lambda^{k+1} \quad := \mathrm{prox}_{M_k^{-1}h}\left(\lambda_k + \frac{1}{M_k}\left(\mathbf{A}\mathbf{x}^*(\lambda^k) - \mathbf{b}\right)\right).$$

- The universal dual accelerated gradient step:

$$\begin{cases} t_k & := 0.5\left(1 + \sqrt{1 + 4t_{k-1}^2}\right) \\ \hat{\lambda}^k & := \bar{\lambda}^k + \frac{t_{k-1}-1}{t_k}\left(\bar{\lambda}^k - \hat{\lambda}^{k-1}\right) \\ \lambda^{k+1} & := \mathrm{prox}_{M_k^{-1}h}\left(\hat{\lambda}^k + \frac{1}{M_k}\left(\mathbf{A}\mathbf{x}^*(\hat{\lambda}^k) - \mathbf{b}\right)\right). \end{cases}$$

Here, h is defined by $h(\lambda) := \sup_{\mathbf{r}\in\mathcal{K}}\langle\lambda, \mathbf{r}\rangle$.

Example: Robust matrix completion with $\approx 1:50$ subsampling

Problem formulation

Let $\Omega \subseteq \{1, \cdots, p\} \times \{1, \cdots, q\}$ be a subset of indexes and $\mathbf{M}_\Omega = (\mathbf{M}_{ij})_{(i,j)\in\Omega}$ be the observed entries of a missed matrix $\mathbf{M}$. $\mathcal{P}_\Omega$ is the projection on the subset Ω.

$$f^\star := \min_{\mathbf{X}\in\mathbb{R}^{p\times q}} \left\{ f(\mathbf{X}) := \frac{1}{2}\|\mathbf{X}\|_\star^2 : \|\mathcal{P}_\Omega(\mathbf{X}) - \mathbf{M}_\Omega\|_1 \leq \tau \right\}.$$

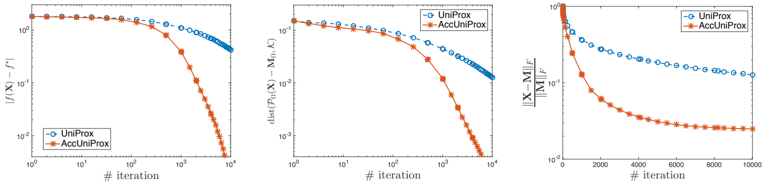


Figure: The performance of UniProx and AccUniProx algorithms.

Setup

- Synthetic data $p = 1000$, $q = 4000$, and rank $r = 6$
- Number of samples $n := |\Omega| = 9 \cdot 10^4$
- Input parameters $\lambda^0 = \mathbf{0}^n$ and $\epsilon = 2 \cdot 10^{-2}$

Example: Nuclear-norm constrained matrix completion - I

Problem formulation

Let $\Omega \subseteq \{1, \cdots, p\} \times \{1, \cdots, q\}$ be a subset of indexes and $\mathbf{M}_\Omega = (\mathbf{M}_{ij})_{(i,j)\in\Omega}$ be the observed entries of a missed matrix $\mathbf{M}$. $\mathcal{P}_\Omega$ is the projection on the subset Ω.

$$f^\star := \min_{\mathbf{X}\in\mathbb{R}^{p\times q}} \left\{ \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{X}) - \mathbf{M}_\Omega\|_F^2 : \|\mathbf{X}\|_\star \leq \varphi^\star \right\}$$

Setup

- Synthetic data of size $p = 400, q = 2000$ with rank $r = 10$.
- Number of samples $n := |\Omega| = 7.5 \cdot 10^4$.
- $\varphi^\star = \|\mathbf{M}\|_\star$ is assumed to be known.
- Input parameters $\lambda^0 = \mathbf{0}^n$ and $\epsilon = 2 \cdot 10^{-6}$.

Example: Nuclear-norm constrained matrix completion - II

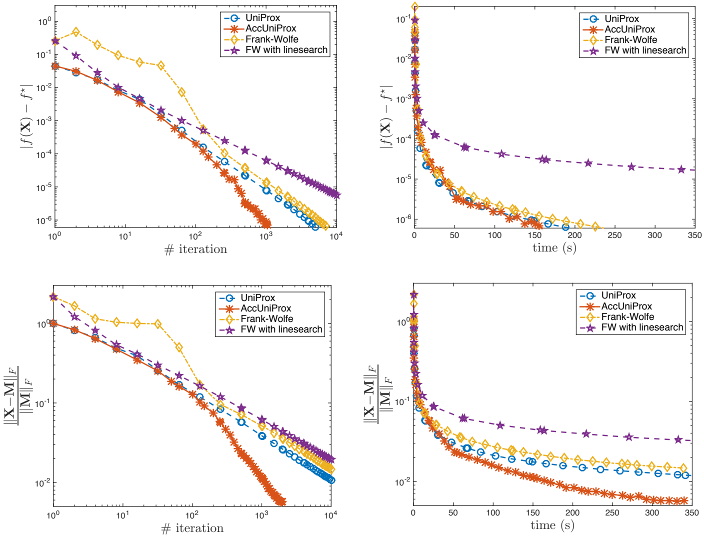


Figure: The performance of (Acc)UniProx and Frank-Wolfe algorithms.

Statistical learning

Problem (Risk minimization for prediction)

Let $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_n, b_n) \in \mathbb{R}^p \times \mathbb{R}$ be i.i.d. random variables with an unknown probability distribution. Let $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a given loss function, and $\mathcal{F}$ be a set of prediction rules $f_{\mathbf{x}}$ parameterized by $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p$. Find a vector $\mathbf{x}^\star \in \mathcal{X}$ such that

$$R(\mathbf{x}^\star) := \mathbb{E}_{\mathbf{a},b}\mathcal{L}(b, f_{\mathbf{x}^\star}(\mathbf{a})) \qquad \text{(risk)}$$

is small. Let $\mathbf{x}^\natural$ be the true minimizer of R.

Optimization formulation (Empirical risk minimization)

The risk R is not tractable because the distribution of $(\mathbf{a}, b)$ is unknown. We consider minimizing the *empirical risk*:

$$\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ R_n(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(b_i, f_{\mathbf{x}}(\mathbf{a}_i)) \right\}.$$

- Notice that we can only obtain a numerical approximation $\hat{\mathbf{x}}_\epsilon$ of $\mathbf{x}^\star$ such that

$$R_n(\hat{\mathbf{x}}_\epsilon) \leq R_n(\mathbf{x}^\star) + \epsilon_n,$$

for some $\epsilon_n > 0$.

Statistical learning contd.

Fact: Uniform law of large numbers (cf. [19] for details)

Under some conditions, when $n \to \infty$,

$$\mathbb{E} \sup_{\mathbf{x} \in \mathcal{X}} \{|R(\mathbf{x}) - R_n(\mathbf{x})|\} := \rho_n \to 0.$$

Performance of Statistical Learning [5]

Recall that $\hat{\mathbf{x}}_\epsilon$ is a numerical approximation of $\mathbf{x}^\star$ and that $\mathbf{x}^\natural$ is the true minimizer of R on $\mathcal{X}$. Then, the excess risk satisfies

$$\begin{aligned} R(\hat{\mathbf{x}}_\epsilon) - R(\mathbf{x}^\natural) &= \mathbb{E}\left[R(\hat{\mathbf{x}}_\epsilon) - R_n(\hat{\mathbf{x}}_\epsilon)\right] + \mathbb{E}\left[R_n(\hat{\mathbf{x}}_\epsilon) - R_n(\mathbf{x}^\star)\right] + \\ &\quad \mathbb{E}\left[R_n(\mathbf{x}^\star) - R_n(\mathbf{x}^\natural)\right] + \mathbb{E}\left[R_n(\mathbf{x}^\natural) - R(\mathbf{x}^\natural)\right] \\ &\leq \rho_n + \epsilon_n + 0 + \rho_n. \end{aligned}$$

A stylized formalization of the time-data tradeoff

$$\underbrace{R(\hat{\mathbf{x}}_\epsilon) - R(\mathbf{x}^\natural)}_{\text{learning quality}} \leq \underbrace{\epsilon_n}_{\text{needs "time" } t(k)} + 2 \underbrace{\rho_n}_{\text{needs "data"} n} \leq \varepsilon$$

Stochastic convex optimization: General setting

Data science applications often involve to solve the following **stochastic problem**:

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ F(\mathbf{x}) := \mathbb{E}_\xi\left\{\hat{f}(\mathbf{x},\xi)\right\} + g(\mathbf{x}) \right\},$$

where $\hat{f} : \mathbb{R}^p \times \Omega \mapsto \mathbb{R}$ is such that for a fixed realization $\xi \in \Omega$ of the random variable ξ, $\hat{f}(\cdot,\xi)$ is convex, g is also a convex function (regularization term), and

$$f(\mathbf{x}) := \mathbb{E}_\xi\left\{\hat{f}(\cdot,\xi)\right\} := \int_\Omega f(\cdot,\xi)\,dP(\xi)$$

is the **expectation** over ξ with a given distribution P on its support Ω.

Basic assumptions

- One can generate iid samples $\{\xi_i\}_{i\geq 0}$ of ξ.
- Given $(\mathbf{x},\xi)$, one can evaluate $\nabla\hat{f}(\mathbf{x},\xi)$: $\nabla f(\mathbf{x}) := \mathbb{E}_\xi\left\{\nabla\hat{f}(\mathbf{x},\xi)\right\} \in \partial f(\mathbf{x})$.

Stochastic convex optimization: General setting

Data science applications often involve to solve the following **stochastic problem**:

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ F(\mathbf{x}) := \mathbb{E}_\xi \left\{ \hat{f}(\mathbf{x},\xi) \right\} + g(\mathbf{x}) \right\},$$

where $\hat{f} : \mathbb{R}^p \times \Omega \mapsto \mathbb{R}$ is such that for a fixed realization $\xi \in \Omega$ of the random variable ξ, $\hat{f}(\cdot,\xi)$ is convex, g is also a convex function (regularization term), and

$$f(\mathbf{x}) := \mathbb{E}_\xi \left\{ \hat{f}(\cdot,\xi) \right\} := \int_\Omega f(\cdot,\xi)\, dP(\xi)$$

is the **expectation** over ξ with a given distribution P on its support Ω.

Basic assumptions

- One can generate iid samples $\{\xi_i\}_{i\geq 0}$ of ξ.
- Given $(\mathbf{x},\xi)$, one can evaluate $\nabla\hat{f}(\mathbf{x},\xi)$: $\nabla f(\mathbf{x}) := \mathbb{E}_\xi \left\{ \nabla\hat{f}(\mathbf{x},\xi) \right\} \in \partial f(\mathbf{x})$.

Two strategies: Monte-Carlo methods

- **Stochastic approximation (SA):** Robbins-Monro [56], Polyak and Juditsky [55], Nemirovskii *et al.* [44], Lan [35], etc.
- **Empirical risk minimization (ERM):** Let $\{\xi_i\}_{i=1}^n$ is n-iid samples of ξ. Then:

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^n \hat{f}(\mathbf{x},\xi_i) + g(\mathbf{x}) \right\}.$$

Stochastic approximation algorithms

The classical stochastic approximation (SA) algorithm

Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and compute:

$$\boxed{\mathbf{x}^{k+1} := \mathrm{prox}_{\alpha_k g}\left(\mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \xi_k)\right),}$$

where $G(\mathbf{x}^k, \xi^k)$ is a stochastic subgradient of $\hat{f}$ at $(\mathbf{x}^k, \xi_k)$ and $\alpha_k > 0$ is a step-size.

Convergence [44]: $\mathcal{O}(1/k)$ under the assumptions of Lipschitz gradient, strongly convex and $\mathbb{E}\left\{\|G(\mathbf{x}, \xi)\|_2^2\right\} \leq M$ for all $\mathbf{x} \in \mathrm{dom} F$.

A robust SA algorithm [44]

- **Main steps:** one averaging step and one stochastic gradient step

$$\bar{\mathbf{x}}^k := \Big(\sum_{i=0}^k \alpha_i\Big)^{-1} \sum_{i=0}^k \alpha_i \mathbf{x}^i, \quad \text{and} \quad \mathbf{x}^{k+1} := \mathrm{prox}_{\alpha_k g}\left(\mathbf{x}^k - \alpha_k G(\mathbf{x}^k, \xi_k)\right).$$

- **Step-size selection:** There are two strategies
  1. **Constant step:** $\alpha_k := \frac{D}{M\sqrt{k_{\max}}}$, where D is the diameter of $\mathrm{dom} F$.
  2. **Varying step:** $\alpha_k := \frac{\theta D}{M\sqrt{k}}$.
- **Convergence rate:**

$$\mathbb{E}\left\{F(\bar{\mathbf{x}}^k) - F^\star\right\} \leq \mathcal{O}(1)\frac{DM}{\sqrt{k}}.$$

The empirical risk minimization

The composite empirical risk minimization (ERM) formulation

In machine learning and data sciences, we are interested in the following composite empirical risk minimization problem:

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) + g(\mathbf{x}) \right\}.$$

where n is the number of data points $(n \gg 1)$, f_i is convex for $i = 1, \ldots, n$ and g is a convex regularizer.

Here: $f_i(\mathbf{x}) = \hat{f}(\mathbf{x}, \xi_i)$ is a realization of a function $f(\mathbf{x}, \cdot)$ of the random variable ξ.

Common settings

Loss functions:

- Least squares: $f_i(\mathbf{x}) := \frac{1}{2}(\mathbf{a}_i^T\mathbf{x} - b_i)^2$
- Hinge loss: $f_i(\mathbf{x}) := \max\left\{0, 1 - b_i\mathbf{a}_j^T\mathbf{x}\right\}$
- Logistic loss: $f_i(\mathbf{x}) := \log(1 + \exp(b_i\mathbf{a}_i^T\mathbf{x}))$.

Regularizers:

- $g(\mathbf{x}) := \rho\left\|\mathbf{x}\right\|_1$ or $g(\mathbf{x}) := \frac{\rho}{2}\left\|\mathbf{x}\right\|_2^2$ for given $\rho = 0$.
- $g(\mathbf{x}) := \iota_{\mathcal{X}}(\mathbf{x})$ - the indicator function of a convex set $\mathcal{X}$.

Stochastic gradient descent for ERM: A review

Empirical risk minimization

Consider the **non-composite** problem

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right\}.$$

where $f_i : \mathbb{R}^p \to \mathbb{R}$ is convex.

The big data case: n **is very very big**.

Deterministic vs stochastic approach

- **Standard gradient algorithms:** Each iteration, it performs

$$\mathbf{x}^{k+1} := \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k),$$

which requires a full gradient of f

$$\nabla f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}).$$

  - It needs n **single gradient** term $\nabla f_i(\mathbf{x})$ at each iteration k.
  - Quasi-Newton methods still require $O(n)$.
  - Convergence with constant α_k or line-search.

Deterministic vs stochastic approach (cont.)

- A **simple stochastic method** operates on a **single** $\nabla f_i(\mathbf{x})$ **term** at each iteration k

$$\mathbf{x}^{k+1} := \mathbf{x}^k - \alpha_k \nabla f_{i_k}(\mathbf{x}^k),$$

where $i_k \in \{1, \ldots, n\}$ is a random index to select **one** component of f.

  - Gives unbiased estimate of true gradient $\nabla f(\mathbf{x})$

$$\mathbb{E}[\underbrace{\nabla f_{i_k}(\mathbf{x})}_{G(\mathbf{x}, i_k)}] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

  - The computation of a stochastic approximate gradient is independent of n.
  - As in subgradient method, we require $\alpha_k \to 0$.
  - Classical choice is $\alpha_k = \mathcal{O}(1/k)$.

Observations:

  - It is n times cheaper per-iteration than the standard gradient method
  - It attempts to minimize the risk (e.g., in data streaming) to obtain $\mathbf{x}^\natural$
  - It can approximately solve a deterministic optimization problem (i.e., $\mathbf{x}^\natural = \mathbf{x}^\star$)

Convex optimization zoo

- **Convergence rate** of **subgradient methods** in the nonsmooth case:
  - **Deterministic subgradient method:** $\mathcal{O}(1/\sqrt{k})$.
  - **Stochastic subgradient method:** $\mathcal{O}(1/\sqrt{k})$.

They are both the same up to constants, which can be large.

Observation: Stochastic iterations are n times faster, but how many iterations?

- Convergence rates under different assumptions.

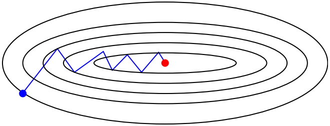
Algorithm	Assumptions	Exact	Stochastic
Subgradient	Convex	$\mathcal{O}(1/\sqrt{k})$	$\mathcal{O}(1/\sqrt{k})$
Subgradient	Strongly convex	$\mathcal{O}(1/k)$	$\mathcal{O}(1/k)$

- Good news for non-smooth problems:
  - stochastic algorithms can be as fast as deterministic ones
- We can solve non-smooth problems n times faster!
- Bad news for smooth problems:
  - smoothness does not necessarily help stochastic methods.

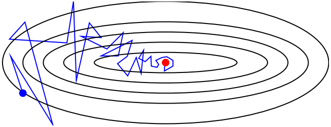
Algorithm	Assumptions	Exact	Stochastic
Gradient	Convex	$\mathcal{O}(1/k)$	$\mathcal{O}(1/\sqrt{k})$
Gradient	Strongly convex	$\mathcal{O}((1-\mu/L)^k)$	$\mathcal{O}(1/k)$

Stochastic vs. Deterministic Gradient Methods

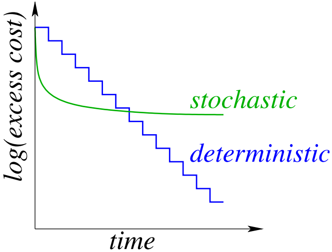
- Deterministic gradient method [Cauchy, 1847]:



- Stochastic gradient method [Robbins & Monro, 1951]:



- Plot of **convergence rates** in smooth/strongly convex case:



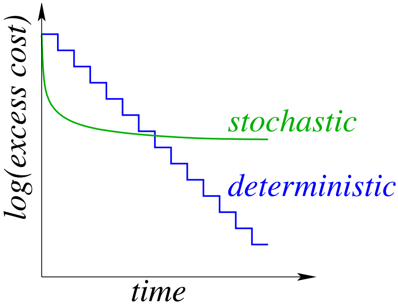
Speeding up stochastic gradient methods

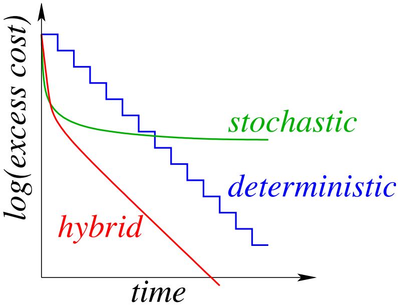
- **Improving performance:** We can try accelerated/Newton-like stochastic methods:
  - These do not improve on the worst-case convergence rates.
  - But may improve performance at start if noise is small.
- **Improving speed:**

  Large step-size:
  - Choose $\alpha_k = \mathcal{O}(1/k^\gamma)$ for $\gamma \in (0.5, 1)$ [Moulines& Bach, 2011]: more robust than $\mathcal{O}(1/k)$.
  - Constant step-size α achieves rate of $\mathcal{O}(\rho^k) + \mathcal{O}(\alpha)$ [Nedic & Bertsekas, 2000].

  Averaging:
  - Gradient averaging improves constants ('dual averaging'), see [Nesterov, 2009].
  - Averaging in smooth cases achieves the same asymptotic rate as optimal stochastic Newton method [Polyak & Juditsky, 1992].
  - Averaging and constant step-size achieves $\mathcal{O}(1/k)$ rate for stochastic Newton-like methods without strong convexity [Bach & Moulines, 2013].

Motivation for hybrid methods for smooth problems





Convex optimization zoo

Convergence rate

Algorithm	Rate	Grads
Stochastic Gradient	$\mathcal{O}(1/k)$	1
Gradient	$\mathcal{O}((1-\mu/L)^k)$	N
Stochastic averaging gradient (SAG)	$\mathcal{O}((1-\min\{\frac{\mu}{16L_i}, \frac{1}{8n}\})^k)$	1

- L_i is the Lipschitz constant over all ∇f_i ($L_i \geq L$).
- SAG has a similar speed to the gradient method, but only looks at one training example per iteration [Le Roux et al., 2012].

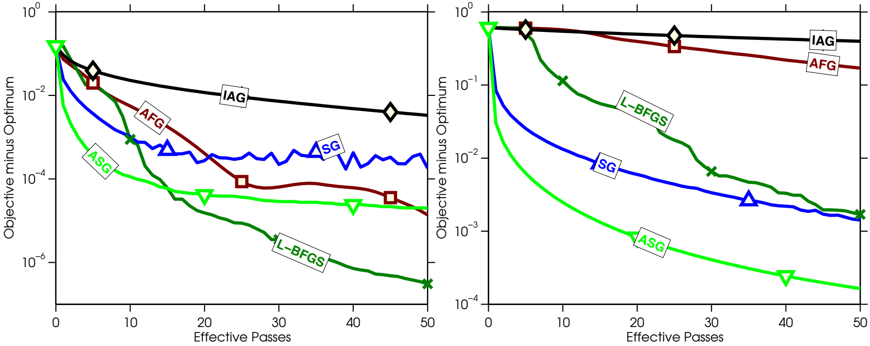
Extensions

Recent work extends this result in various ways:

- Similar rates for stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
- Memory-free variants [Johnson & Zhang, 2013; Madavi et al., 2013]
- Proximal-gradient variants [Mairal, 2013]
- ADMM variants [Wong et al., 2013]
- Improved constants [Defazio et al., 2014]
- Non-uniform sampling [Schmidt et al., 2013]

Comparing full gradient and stochastic gradient methods

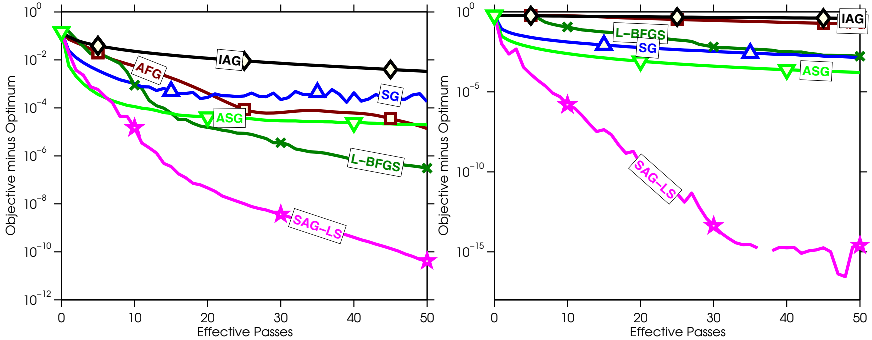
- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



- IAG = incremental aggregated gradient [Blatt et al, 2007]
- AFG = accelerated full gradient [Nesterov, 1983]
- SG = stochastic gradient [Robbins & Monro, 1951]
- ASG = average stochastic gradient [Nemirovskii et al, 2009]
- L-BFGS = limited memory BFGS

Comparing full gradient and stochastic gradient methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



- IAG = incremental aggregated gradient [Blatt et al, 2007]
- AFG = accelerated full gradient [Nesterov, 1983]
- SG = stochastic gradient [Robbins & Monro, 1951]
- ASG = average stochastic gradient [Nemirovskii et al, 2009]
- L-BFGS = limited memory BFGS
- SAG-LS = stochastic average gradient with line-search [Schmidt et al., 2013]

Coordinate gradient descent methods

Problem setting

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

- $f : \mathbb{R}^p \to \mathbb{R}$ is convex and smooth, where $p \gg 1$ is **relative large**.

Assumptions

- **Block decomposition:** $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_m]$, where $\mathbf{x}_i \in \mathbb{R}^{p_i}$ such that $\sum_{i=1}^m p_i = p$.
- **Coordinate Lipschitz gradient:** $\nabla_i f(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}_i}(\mathbf{x}) = \mathbf{U}_i^T \nabla f(\mathbf{x})$ is L_i-Lipschitz continuous, i.e.:

$$\|\nabla_i f(\mathbf{x} + \mathbf{U}_i\mathbf{d}_i) - \nabla_i f(\mathbf{x})\|_* \le L_i \|\mathbf{d}_i\|, \ \forall \mathbf{d}_i \in \mathbb{R}^{p_i}, \ \mathbf{x} \in \mathbb{R}^p,$$

where $\mathbf{U}_i \in \mathbb{R}^{p\times p_i}$ such that $\mathbb{I} = [\mathbf{U}_1, \ldots, \mathbf{U}_m]$ the identity matrix in $\mathbb{R}^p$[8].

Sharp operator

For any norm $\|\cdot\|$, we recall the sharp-operator: $[\mathbf{x}]^\sharp := \arg\max_{\mathbf{z}}\{\langle \mathbf{x}, \mathbf{z}\rangle - (1/2)\|\mathbf{z}\|^2\}$.

[8] $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$

Coordinate gradient descent algorithm

Conceptual coordinate descent method (CDM) [47]

- **Initialization:** Choose $\mathbf{x}^0 \in \mathbb{R}^p$.
- **Iteration:** For $k = 0, \cdots, k_{\max}$, perform:
  1. Choose a coordinate $i_k \in \{1, \ldots, m\}$ a priori.
  2. Update:
  $$\mathbf{x}^{k+1} := \mathbf{x}^k - \alpha_k [\nabla_i f(\mathbf{x})]^\sharp,$$
  where $\alpha_k > 0$ is a given step-size and $[\cdot]^\sharp$ is the sharp-operator.

Three unspecified steps:

1. The choice of norms: Depending on the function f (e.g., L_i)
   - weighted or without weighted Euclidian norms
   - a general ℓ_p-norm.
2. The choice of coordinate i_k: There are many possibilities, e.g.,
   - sequential: Based on $|\nabla_i f(\mathbf{x}^k)|$, which requires the m-terms $\nabla_i f$.
   - Cycling: Hard to prove convergence.
   - Random: Convergence on the expectation or probability.
3. The choice of step-size α_k: There are at least two strategies
   - constant step-size: Depending on, e.g, Lipschitz constant.
   - adaptive step-size: Often work better.

Different strategies

Randomized coordinate descent method (Nesterov's method)

- Choose i_k based on $P\{i_k = j\} = \frac{L_j^{\beta}}{\sum_{l=1}^{m} L_l^{\beta}}$, for $j \in \{1, \cdots, m\}$ and $\beta \geq 0$.
- If $\beta = 0$, i.e. $P\{i_k = j\} = \frac{1}{m}$ for $j \in \{1, \cdots, m\}$, it is uniform random coordinate descent.

Theorem: Convergence guarantees

Let $\{\mathbf{x}^k\}$ be the sequence generated by CDM using Nesterov's strategy. Then:

$$\mathbb{E}\{f(\mathbf{x}^k)\} - f^{\star} \leq \frac{2}{k+4}\left[\sum_{j=1}^{m} L_j^{\beta}\right] R_{1-\beta}^2(\mathbf{x}^0),$$

where $R_{\gamma}(\mathbf{x}^0) := \max_{\mathbf{x}} \left\{ \left[\sum_{j=1}^{m} L_j^{\gamma} \left\|\mathbf{x}_j^0 - \mathbf{x}_j^{\star}\right\|^2 \right]^{1/2} : f(\mathbf{x}) \leq f(\mathbf{x}^0) \right\}$.

With $\beta = 0$, i.e. using uniform random strategy, we have:

$$\mathbb{E}\{f(\mathbf{x}^k)\} - f^{\star} \leq \frac{2m}{k+4}\left[\sum_{j=1}^{m} L_j^{\beta}\right] R_1^2(\mathbf{x}^0) \quad \Longrightarrow \mathcal{O}\left(\frac{m}{k}\right).$$

Convergence rate of coordinate descent: Details

- **Step-size strategies**
  - The *steepest descent* choice is $j = \arg\max_j\{|\nabla_j f(\mathbf{x})|\}$ (Gauss-Southwell).
  - Convergence rate (strongly-convex, partials are L_j-Lipschitz):

$$\mathcal{O}((1 - \mu/L_j D)^t).$$

  - L_j is typically much smaller than L across all coordinates:
    - Coordinate descent is faster if we can do D coordinate descent steps for cost of one gradient step.

- **Extension and improvements: various**. Here are few examples:
  - Projected coordinate descent (product constraints) [Nesterov, 2010; Beck, 2013]
  - Proximal coordinate descent (separable non-smooth term) [Richtarik & Takac, 2011]
  - Composite convex settings and linear constraints [Necoara et al, 2012; Necoara & Patrascu, 2014; Beck, 2014].
  - Frank-Wolfe coordinate descent (product constraints) [LaCoste-Julien et al., 2013]
  - Accelerated version [Nesterov, 2010; Fercoq & Richtarik, 2013]
  - Parallel variants [Richtarik & Takac, 2012; Necoara & Clipici, 2013]

Randomized linear algebra

Problem setting

Consider problems of the form

$$\min_{\mathbf{x}} f(\mathbf{A}\mathbf{x})$$

where bottleneck is matrix multiplication and $\mathbf{A}$ is low-rank.

Randomized linear algebra techniques

- Randomized linear algebra approaches uses

$$\mathbf{A} \approx \mathbf{Q}(\mathbf{Q}^T\mathbf{A}),$$

  or choses random row/columns subsets.
- Now, we work with $f(\mathbf{Q}(\mathbf{Q}^T\mathbf{A})\mathbf{x})$ instead of $f(\mathbf{A}\mathbf{x})$.
- **When does it work well?** $\mathbf{Q}$ formed from Gram-Schmidt and matrix multiplication with random vectors gives very good approximation bounds, if singular values decay quickly. [Halko et al., 2011; Mahoney, 2011]
- But, it may work quite badly if singular values decay slowly.

Motivation for parallel and distributed optimization

Motivation

We must use parallel and/or distributed computation.

- We can not make large gains in serial computation speed.
- Datasets no longer fit on a single machine.

Two major issues:

- Synchronization: we cannot wait for the slowest machine.
- Communication: we cannot transfer all information.

Embarrassing parallelism

- Many optimization problems are embarrassingly parallel: Split tasks across m machines, solve independently, combine.
- Decomposition methods we discussed readily fit into this setting.
- E.g., computing the gradient in deterministic gradient method,

$$\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x}) = \frac{1}{n}\Big(\sum_{i=1}^{n/m}\nabla f_i(\mathbf{x}) + \sum_{i=(n/m)+1}^{2n/m}\nabla f_i(\mathbf{x}) + \dots\Big).$$

- These allow optimal linear speedups: You should always consider this first!

Asynchronous computation

Asynchronous computation

- Do we have to wait for the last computer to finish? **NO!**
- **HOW?**: Updating asynchronously saves a lot of time.
- One posibility: stochastic gradient method on shared memory

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f_{i_k}(\mathbf{x}_{k-\tau}).$$

- You need to decrease step-size in proportion to asynchrony.
- Convergence rate decays gracefully with delay τ [Niu et al., 2011].

Reduced communication

Parallel coordinate descent

- It may be expensive to communicate $\mathbf{x}$.
- **One possibility:** using parallel coordinate descent

$$\mathbf{x}_{j_l} = \mathbf{x}_{j_l} - \alpha_{j_l} \nabla_{j_l} f(\mathbf{x}), \quad l = 1, \ldots, r.$$

Here, we only need to communicate single coordinates, and need to decrease step-size for convergence.
Speedup is based on density of graph [Richtarik & Takac, 2013].

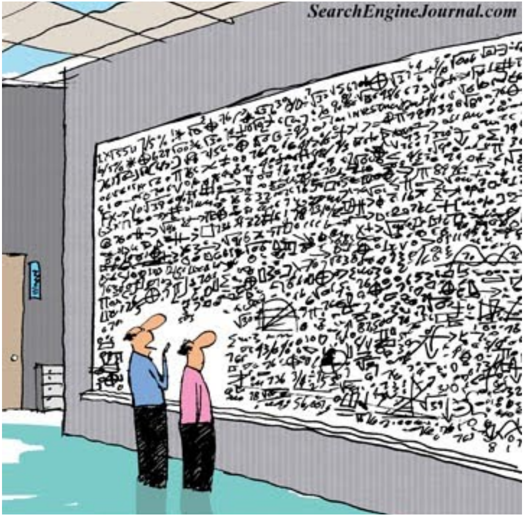
Decentralized gradient

- We may need to **distribute** the data across machines, but may not want to update a "centralized" vector $\mathbf{x}$.
- **One possibility:** decentralized gradient method
  - Each processor has its own: data samples $f_1, f_2, \ldots f_m$ and vector $\mathbf{x}_c$.
  - Each processor only communicates with a limited number of neighbors nei(c).

$$\mathbf{x}_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} \mathbf{x}_c - \frac{\alpha_c}{m} \sum_{i=1}^{m} \nabla f_i(\mathbf{x}_c).$$

- Gradient descent is special case where all neighbors communicate.
- With modified update, rate decays gracefully as graph becomes sparse [Shi et al., 2014].
- Can also consider communication failures [Agarwal & Duchi, 2011].

Conclusions



And, this is how you solve Big Data problems...

References

[1] Ben Adcock, Anders C. Hansen, Clarice Poon, and Bogdan Roman.
Breaking the coherence barrier: A new theory for compressed sensing.
http://arxiv.org/abs/1302.0561, Feb. 2013.

[2] Richard G. Baraniuk, Volkan Cevher, Marco F. Duarte, and Chinmay Hegde.
Model-based compressive sensing.
IEEE Trans. Inf. Theory, 56(4):1982–2001, April 2010.

[3] Heinz H. Bauschke and Patrick L. Combettes.
Convex analysis and monotone operator thoery in Hilbert spaces.
Springer, New York, NY, 2011.

[4] Amir Beck and Marc Teboulle.
A fast iterative shrinkage-thresholding algorithm for linear inverse problems.
SIAM J. Imaging Sci., 2(1):183–202, 2009.

[5] Léon Bottou and Oliver Bousquet.
The tradeoffs of large scale learning.
In *Advances in Neural Information Processing Systems*, 2007.

[6] Luis M Briceno-Arias and Patrick L Combettes.
A monotone+ skew splitting model for composite monotone inclusions in duality.
SIAM Journal on Optimization, 21(4):1230–1250, 2011.

[7] John J. Bruer, Joel A. Tropp, Volkan Cevher, and Stephen R. Becker.
Time-data tradeoffs by smoothing.
submitted for publication, 2014.

[8] Antonin Chambolle and Thomas Pock.
A first-order primal-dual algorithm for convex problems with applications to imaging.
J. Math. Imaging Vis., 40:120–145, 2011.

[9] Venkat Chandrasekaran and Michael I. Jordan.
Computational and statistical tradeoffs via convex relaxation.
Proc. Natl. Acad. Sci., 110(13):E1181–E1190, 2013.

[10] G. Chen and M. Teboulle.
A proximal-based decomposition method for convex minimization problems.
Math. Program., 64:81–101, 1994.

References

[11] P. L. Combettes and V. R. Wajs.
Signal recovery by proximal forward-backward splitting.
Multiscale Model. Simul., 4:1168–1200, 2005.

[12] Patrick L. Combettes and Jean-Christophe Pesquet.
Proximal splitting methods in signal processing.
In Heinz Bauschke, Regina S. Burachik, Patrick Combettes, Veit Elser, D. Russell Luck, and Henry Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter 10. Springer, New York, NY, 2011.

[13] I. Daubechies, M. DeFriese, and C. DeMol.
An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.
Commun. Pure Appl. Math., 57:1413—1457, 2004.

[14] D. Davis.
Convergence rate analysis of the forward-Douglas-Rachford splitting scheme.
UCLA CAM report 14-73, 2014.

[15] D. Davis and W. Yin.
Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions.
UCLA CAM report 14-58, 2014.

[16] D. Davis and W. Yin.
A three-operator splitting scheme and its optimization applications.
Tech. Report., 2015.

[17] JE Dennis and Jorge J Moré.
A characterization of superlinear convergence and its application to quasi-newton methods.
Mathematics of Computation, 28(126):549–560, 1974.

[18] Marco F. Duarte, Dharmpal Davenport, Mark A. adn Takhar, Jason N. Laska, Ting Sun, Kevin F. Kelly, and Richard G. Baraniuk.
Single-pixel imaging via compressive sampling.
IEEE Sig. Process. Mag., 25(2):83–91, March 2008.

[19] R. M. Dudley.
Uniform Central Limit Theorems.
Cambridge Univ. Press, New York, NY, second edition, 2014.

References

[20] Jonathan Eckstein and Dimitri P. Bertsekas.
On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators.
Math. Program., 55:293–318, 1992.

[21] Marwa El Halabi and Volkan Cevher.
A totally unimodular view of structured sparsity.
In *18th Int. Conf. Artificial Intelligence and Statistics*, 2015.

[22] J. E. Esser.
Primal-dual algorithm for convex models and applications to image restoration, registration and nonlocal inpainting.
Phd. thesis, University of California, Los Angeles, Los Angeles, USA, 2010.

[23] M. Figueiredo and J. Bioucas-Dias.
Restoration of Poissonian images using alternating direction optimization.
IEEE Transactions on Image Processing, 19:3133–3145, 2010.

[24] M. Figueiredo, J. Bioucas-Dias, and R. Nowak.
Majorization-minimization algorithms for wavelet-based image restoration.
IEEE Transactions on Image Processing, 16:2980–2991, 2007.

[25] M. Figueiredo and R. Nowak.
An em algorithm for wavelet-based image restoration.
IEEE Transactions on Image Processing, 12:906–916, 2007.

[26] D. Gabay and B. Mercier.
A dual algorithm for the solution of nonlinear variational problems via finite element approximation.
Computers & Mathematics with Applications, 2(1):17 – 40, 1976.

[27] P. Giselsson and S. Boyd.
Monotonicity and Restart in Fast Gradient Methods.
In *IEEE Conference on Decision and Control*, Los Angeles, USA, December 2014. CDC.

[28] T. Goldstein, E. Esser, and R. Baraniuk.
Adaptive Primal-Dual Hybrid Gradient Methods for Saddle Point Problems.
Tech. Report., http://arxiv.org/pdf/1305.0546v1.pdf:1–26, 2013.

[29] T. Goldstein, B. ODonoghue, and S. Setzer.
Fast Alternating Direction Optimization Methods.
SIAM J. Imaging Sci., 7(3):1588–1623, 2012.

References

[30] B. He and X. Yuan.
Convergence analysis of primal-dual algorithms for saddle-point problem: from contraction perspective.
SIAM J. Imaging Sciences, 5:119–149, 2012.

[31] Bingsheng He and Xiaoming Yuan.
On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method.
SIAM J. Numer. Anal., 50(2):700–709, 2012.

[32] Martin Jaggi.
Revisiting Frank-Wolfe: Projection-free sparse convex optimization.
In *Proc. 30th Int. Conf. Machine Learning*, 2013.

[33] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach.
Proximal methods for hierarchical sparse coding.
J. Mach. Learn. Res., 12:2297–2334, 2011.

[34] B. Krishnapuram, M. Figueiredo, L. Carin, and H. Hartemink.
Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds.
IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 27:957–968, 2005.

[35] Guanghui Lan.
An optimal method for stochastic composite optimization.
Math. Program., Ser. A, 133:365–397, 2012.

[36] Lucien Le Cam.
Asymptotic methods in Statistical Decision Theory.
Springer-Verl., New York, NY, 1986.

[37] S. Lefkimmiatis and M. Unser.
Poisson Image Reconstruction with Hessian Schatten-Norm Regularization.
EEE Trans. Image Processing, 22(11):4314–4327, 2013.

[38] Yen-Huan Li, Ya-Ping Hsieh, and Volkan Cevher.
A geometric view on constrained M-estimators.
EPFL-REPORT-205083, École Polytechnique Fédérale de Lausanne, 2015.

[39] Yen-Huan Li, Jonathan Scarlett, Pradeep Ravikumar, and Volkan Cevher.
Sparsistency of ℓ_1-regularized M-estimators.
In *Proc. 18th Inf. Conf. Artificial Intelligence and Statistics*, pages 644–652, 2015.

References

[40] J.-J. Moreau.
Proximité et dualité dans un espace hilbertien.
Bull. Societé Mathematique de France, 93:273—-299, 1965.

[41] I. Necoara and J.A.K. Suykens.
Interior-point lagrangian decomposition method for separable convex optimization.
J. Optim. Theory and Appl., 143(3):567–588, 2009.

[42] Ion Necoara and Andrei Patrascu.
Iteration complexity analysis of dual first order methods for convex programming.
arXiv preprint arXiv:1409.1462, 2014.

[43] Sahand N. Negahban, Pradeep Ravikumar, Martin J. Wainwright, and Bin Yu.
A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers.
Stat. Sci., 27(4):538–557, 2012.

[44] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro.
Robust stochastic approximation approach to stochastic programming.
SIAM J. Optim., 19(4):1574–1609, 2009.

[45] A. S. Nemirovsky and D. B. Yudin.
Problem complexity and method efficiency in optimization.
John Wiley & Sons, Chichester, 1983.

[46] Y. Nesterov.
Introductory lectures on convex optimization: A basic course, volume 87.
Springer, 2004.

[47] Yu. Nesterov.
Efficiency of coordinate descent methods on huge-scale optimization problems.
SIAM J. Optim., 22(2):341–362, 2012.

[48] Yu Nesterov.
Universal gradient methods for convex optimization problems.
Math. Program., Ser. A, 2014.

[49] Yu. E. Nesterov.
A method of solving a convex programming problem with convergence rate $O(1/k^2)$.
Soviet Math. Dokl., 27(2):372–376, 1983.

References

[50] J. Nocedal and S.J. Wright.
Numerical Optimization.
Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[51] B. O'Donoghue and E. E. Candes.
Adaptive Restart for Accelerated Gradient Schemes.
In *Foundations of Computational Mathematics*, pages 1–18, 2013.

[52] Y. Ouyang, Y. Chen, G. LanG. Lan., and E. JR. Pasiliao.
An accelerated linearized alternating direction method of multiplier.
Tech, 2014.

[53] Samet Oymak, Christos Thrampoulidis, and Babak Hassibi.
Simple bounds for noisy linear inverse problems with exact side information.
2013.
arXiv:1312.0641v2 [cs.IT].

[54] N. Parikh and S. Boyd.
Proximal algorithms.
Foundations and Trends in Optimization, 1(3):123–231, 2013.

[55] Boris T Polyak and Anatoli B Juditsky.
Acceleration of stochastic approximation by averaging.
SIAM Journal on Control and Optimization, 30(4):838–855, 1992.

[56] Herbert Robbins and Sutton Monro.
A stochastic approximation method.
The annals of mathematical statistics, pages 400–407, 1951.

[57] R.T. Rockafellar.
Monotone operators and the proximal point algorithm.
SIAM Journal on Control and Optimization, 14:877–898, 1976.

[58] Shai Shalev-Shwartz and Nathan Srebro.
Svm optimization: inverse dependence on training set size.
In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM, 2008.

References

[59] Q. Tran-Dinh and V. Cevher.
Constrained convex minimization via model-based excessive gap.
In *Proc. the Neural Information Processing Systems Foundation conference (NIPS2014)*, pages 1–9, Montreal, Canada, December 2014.

[60] Q. Tran-Dinh and V. Cevher.
A primal-dual algorithmic framework for constrained convex minimization.
2015.
arXiv:1406.5403v2 [math.OC].

[61] Q. Tran-Dinh, A. Kyrillidis, and V. Cevher.
Composite self-concordant minimization.
J. Mach. Learn. Res., 15:374–416, 2015.

[62] Q. Tran-Dinh, Y.-H. Li, and V. Cevher.
Barrier smoothing for nonsmooth convex minimization.
In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 1503–1507, 2014.

[63] Q. Tran-Dinh, I. Necoara, C. Savorgnan, and M. Diehl.
An Inexact Perturbed Path-Following Method for Lagrangian Decomposition in Large-Scale Separable Convex Optimization.
SIAM J. Optim., 23(1):95–125, 2013.

[64] P. Tseng.
Applications of splitting algorithm to decomposition in convex programming and variational inequalities.
SIAM J. Control Optim., 29:119–138, 1991.

[65] A. W. van der Vaart.
Asymptotic Statistics.
Cambridge Univ. Press, Cambridge, UK, 1998.

[66] E. Wei, A. Ozdaglar, and A.Jadbabaie.
A Distributed Newton Method for Network Utility Maximization.
http://web.mit.edu/asuman/www/publications.htm, 2011.

[67] Alp Yurtsever, Quoc Tran-Dinh, and Volkan Cevher.
Universal primal-dual proximal gradient methods.
2015.

References

[68] G. Zhao.
A Lagrangian dual method with self-concordant barriers for multistage stochastic convex programming.
Math. Progam., 102:1–24, 2005.

[69] Peng Zhao, Guilherme Rocha, and Bin Yu.
Grouped and hierarchical model selection through composite absolute penalties.
Department of Statistics, UC Berkeley, Tech. Rep, 703, 2006.