A quest for theoretically-sound optimization

Zev Woodstock

Technische Universität Berlin & Zuse Institute Berlin

February 2024

“How did I get here?”-David Byrne

“How did I get here?”-David Byrne

JAMES MADISON UNIVERSITY

NC STATE UNIVERSITY

TU berlin + ZIB ZUSE INSTITUTE BERLIN

Lise
6.05 – 8 petaflop/second
(roughly 75 – 100 IBM Watsons)

Theoretically-sound optimization

What is optimization?

Optimization in a nutshell ($\mathcal{H} = \mathbb{R}^n$ or any real Hilbert space)

- **Objective function** $f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$.

  e.g., data fidelity in ML, energy efficiency, profit, statistical error, . . .

- An "optimal" $x \in \mathcal{H}$ makes $f(x)$ the smallest or largest

  e.g., minimize error, maximize efficiency

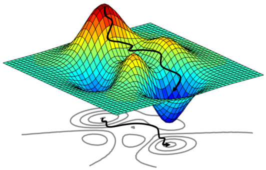


image: towardsdatascience.com

$$\underset{x \in \mathcal{H}}{\text{minimize}} \;\; f(x)$$

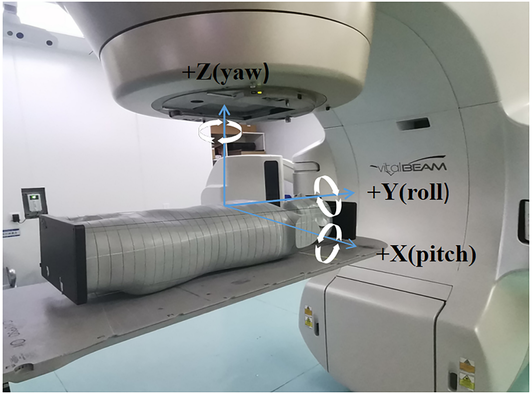
Constraint set(s) $C \subset \mathcal{H}$

e.g., $\mathbb{R}^N_+$, $\mathbb{S}^N_+$, hypercube, solution set of an inverse problem, . . .

$$\iota_C(x) = \begin{cases} 0 & \text{if } \; x \in C \\ +\infty & \text{otherwise.} \end{cases}$$

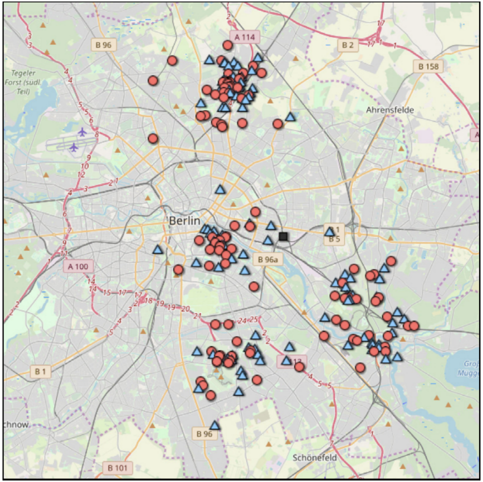
$$\underset{x \in C}{\text{minimize}} \;\; \widetilde{f}(x) = \underset{x \in \mathcal{H}}{\text{minimize}} \;\; \underbrace{\widetilde{f}(x) + \iota_C(x)}_{f}$$

Modeling via optimization



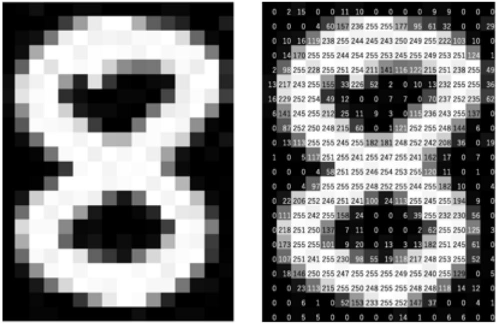
[Torelli et al., *Med. Phys.*, 2023]

image: [Fu et al., *Tech. Cancer Res. Treatment*, 2023]



[Sartori & Buriol, *Comput. Oper. Res.*, 2020]

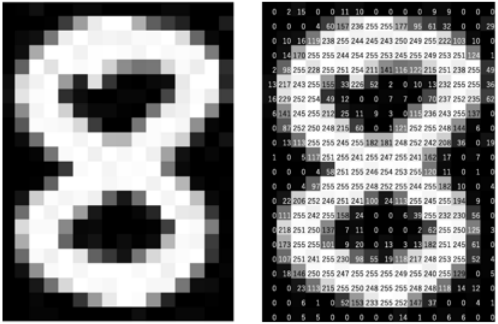
Modeling via optimization



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

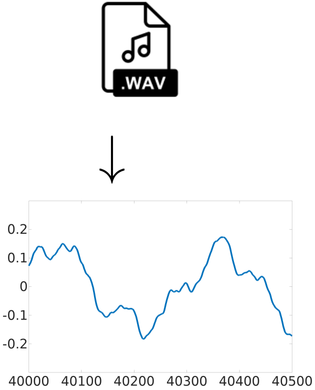
image: `towardsdatascience.com`

Modeling via optimization

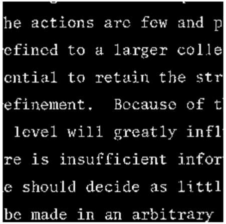


0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

image: towardsdatascience.com

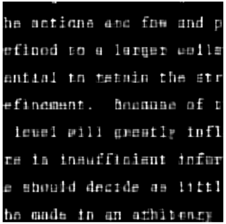


Modeling via optimization



Original

Observation

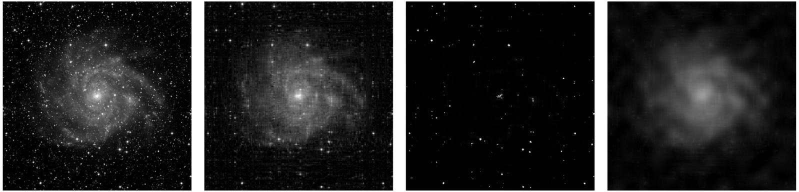


Recovery

[Combettes & ZW., *SIAM J. Imaging Sci.*, 2022]

Modeling via optimization

Original Observation Recovery



Original Observation Recovered stars Recovered galaxy

[Combettes & ZW., *SIAM J. Imaging Sci.*, 2022]

Some fundamental questions

- What are the roadblocks to *provably* solving optimization problems?

  → Nonconvexity, nonsmoothness, and bears – oh my!

Some fundamental questions

- What are the roadblocks to *provably* solving optimization problems?

  → Nonconvexity, nonsmoothness, and bears – oh my!

- What *theoretically-sound* algorithms exist?

Some fundamental questions

- What are the roadblocks to *provably* solving optimization problems?

  → Nonconvexity, nonsmoothness, and bears – oh my!

- What *theoretically-sound* algorithms exist, and can we do better?

  → Splitting, Parallelization, Extrapolation, Asynchronous computation

Theoretically-sound optimization

1. Motivation
2. Background: Theory vs practice
3. Proximity operators: Algorithmic bells and whistles
4. Splitting FW: What if the “usual” tools fail us?
5. More adventuring

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

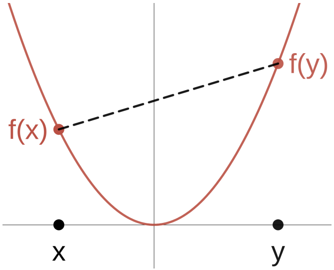
f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

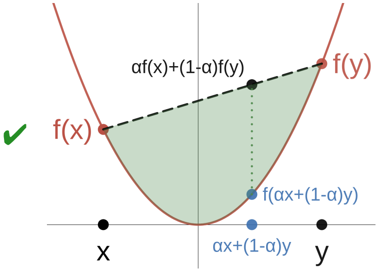


Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

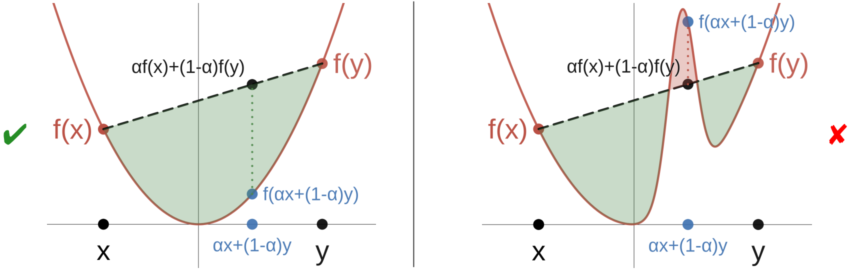


Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

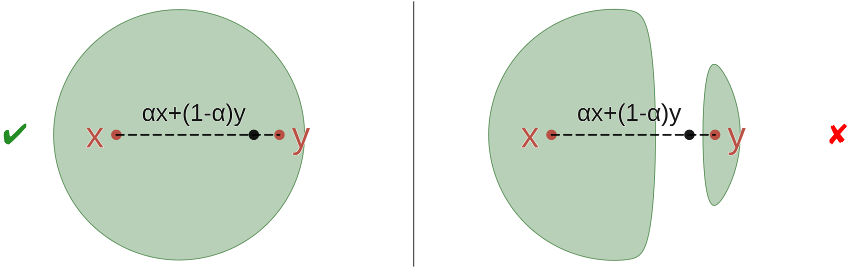


Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

C is **convex** if, for all $x, y \in C$ and $\alpha \in (0, 1)$, $\alpha x + (1 - \alpha) y \in C$.



Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

Let $\Gamma_0(\mathcal{H}) = \{f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} \mid f$ is convex, lower-semicontinuous, and proper$\}$,

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

Let $\Gamma_0(\mathcal{H}) = \{f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} |\ f$ is convex, lower-semicontinuous, and proper$\}$,

e.g., e^x, $-\ln(x)$, $\|\cdot\|^2$, ReLU, Hinge loss, $\|Ax + b\|$, $\|\cdot\|_1$, ι_C (C convex and closed),

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

Let $\Gamma_0(\mathcal{H}) = \{f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} |\ f$ is convex, lower-semicontinuous, and proper$\}$,

e.g., e^x, $-\ln(x)$, $\|\cdot\|^2$, ReLU, Hinge loss, $\|Ax + b\|$, $\|\cdot\|_1$, ι_C (C convex and closed),
$\sup\{f_i | i \in I\}$, affine composition, positive linear combinations, ...

Our setting

$\mathcal{H}$ is a real Hilbert space with inner product $\langle \cdot \mid \cdot \rangle$,
e.g., $\mathbb{R}^n$ with the dot product $\langle x \mid y \rangle = x^T y$.

$\operatorname{Argmin} f = \{x \in \mathcal{H} \mid f(x) = \inf f(\mathcal{H})\}$ is the set of minimizers of f.

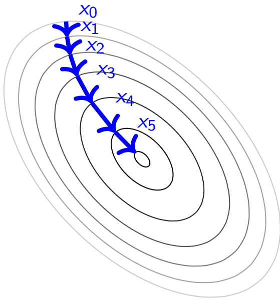
f is **convex** if, for all $x, y \in \mathcal{H}$ and $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y)$

Let $\Gamma_0(\mathcal{H}) = \{f : \mathcal{H} \to \mathbb{R} \cup \{+\infty\} \mid f$ is convex, lower-semicontinuous, and proper$\}$,

e.g., e^x, $-\ln(x)$, $\|\cdot\|^2$, ReLU, Hinge loss, $\|Ax + b\|$, $\|\cdot\|_1$, ι_C (C convex and closed),
sup$\{f_i | i \in I\}$, affine composition, positive linear combinations, ...
not $\|\mathcal{N}(x) - d\|$ for multilayer neural networks

"Traditioooon" -Tevye, Fiddler on the Roof

f is L-**smooth** $(L \geqslant 0)$ if it is differentiable and $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$.



"Traditioooon" -Tevye, Fiddler on the Roof

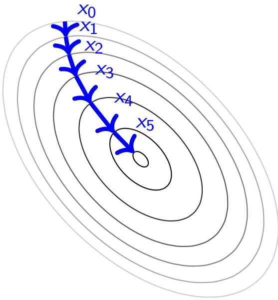
f is L-**smooth** ($L \geqslant 0$) if it is differentiable and $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$.

Gradient Descent

Let $f\colon \mathcal{H} \to \mathbb{R}$ be L-smooth and suppose $\operatorname{Argmin} f \neq \varnothing$. Let $x_0 \in \mathcal{H}$, $\varepsilon > 0$ and for every $n \in \mathbb{N}$, set

$$x_{n+1} = x_n - \lambda_n \nabla f(x_n), \qquad \text{where } \lambda_n \in \left[\varepsilon, \frac{2}{L} - \varepsilon\right] \tag{GD}$$

If $f \in \Gamma_0(\mathcal{H})$, then $(x_n)_{n\in\mathbb{N}}$ converges to a minimizer of f.



Foe #1: Non-convexity

f is L-**smooth** $(L \geqslant 0)$ if it is differentiable and $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$.

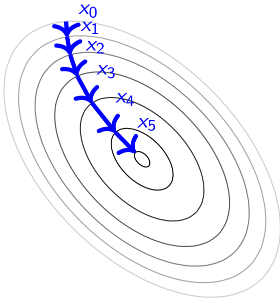
Gradient Descent

Let $f\colon \mathcal{H} \to \mathbb{R}$ be L-smooth and suppose $\operatorname{Argmin} f \neq \varnothing$. Let $x_0 \in \mathcal{H}$, $\varepsilon > 0$ and for every $n \in \mathbb{N}$, set

$$x_{n+1} = x_n - \lambda_n \nabla f(x_n), \qquad \text{where } \lambda_n \in \left[\varepsilon, \frac{2}{L} - \varepsilon\right] \tag{GD}$$

If $f \notin \Gamma_0(\mathcal{H})$, then $(x_n)_{n\in\mathbb{N}}$ coverges to a $\underbrace{\textbf{stationary point}}_{(\nabla f(x^*) = 0)}$.

For x_0 sufficiently close to a minimizer, $(x_n)_{n\in\mathbb{N}}$ converges to one.



"Traditioooon" -Tevye, Fiddler on the Roof

f is L-**smooth** $(L \geqslant 0)$ if it is differentiable and $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x-y\|$.

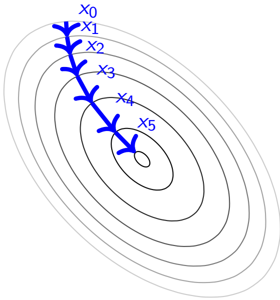
$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

Gradient Descent

Let $f\colon \mathcal{H} \to \mathbb{R}$ be L-smooth and suppose $\operatorname{Argmin} f \neq \varnothing$. Let $x_0 \in \mathcal{H}$, $\varepsilon > 0$ and for every $n \in \mathbb{N}$, set

$$x_{n+1} = x_n - \lambda_n \nabla f(x_n), \qquad \text{where } \lambda_n \in \left[\varepsilon, \frac{2}{L} - \varepsilon\right] \tag{GD}$$

If $f \in \Gamma_0(\mathcal{H})$, then $(x_n)_{n\in\mathbb{N}}$ converges to a minimizer of f.



“Traditioooon” -Tevye, Fiddler on the Roof

f is L-**smooth** ($L \geqslant 0$) if it is differentiable and $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$.

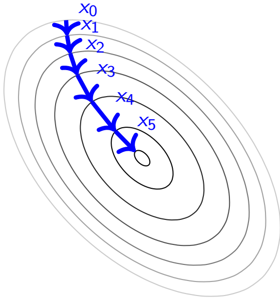
$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

Stochastic Gradient Descent (one variant)

Let $f\colon \mathcal{H} \to \mathbb{R}$ be L-smooth and suppose $\operatorname{Argmin} f \neq \varnothing$. Let $x_0 \in \mathcal{H}$ and for every $n \in \mathbb{N}$, set

$$x_{n+1} = x_n - \frac{1}{n+1}\nabla f_{i_n}(x_n), \qquad \text{where } i_n \sim U(\{1, \dots, m\}) \quad \text{(SGD)}$$

If $f \in \Gamma_0(\mathcal{H})$, then $\mathbb{E}[f(x_n)]$ converges to $\inf_{x \in \mathcal{H}} f(x)$.



Why can't we take the eagles to Mordor?

(A reasonable question to ask, if we didn't read the books)

A common paradigm:

1. Define an objective function
2. Optimize with an efficient algorithm, e.g., SGD with algorithmic differentiation (AD)

Why can't we take the eagles to Mordor?

(A reasonable question to ask, if we didn't read the books)

A common paradigm:

1. Define an objective function
2. Optimize with an efficient algorithm, e.g., SGD with algorithmic differentiation (AD)

[Pontil et al., *Numer. Algorithms*, 2019]

Training a sparse linear binary classifier

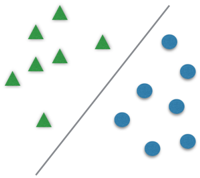


image: adeveloperdiary.com

Why can't we take the eagles to Mordor?

(A reasonable question to ask, if we didn't read the books)

A common paradigm:

1. Define an objective function
2. Optimize with an efficient algorithm, e.g., SGD with algorithmic differentiation (AD)

[Pontil et al., *Numer. Algorithms*, 2019]

Training a sparse linear binary classifier

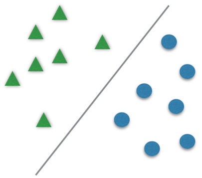


image: adeveloperdiary.com

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \sum_{i\in I_1} \max\{0, 1-\langle x \mid a_i\rangle\} + \sum_{i\in I_2} \max\{0, 1+\langle x \mid a_i\rangle\} + \lambda\|x\|_1$$

Foe #2: Non-differentiability

A common paradigm:

1. Define an objective function
2. Optimize with an efficient algorithm, e.g., SGD with algorithmic differentiation (AD)

Issue: For many objective functions, a gradient does not exist.

[Pontil et al., *Numer. Algorithms*, 2019]

Training a sparse linear binary classifier

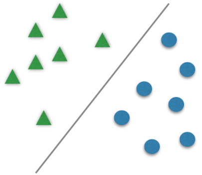


image: adeveloperdiary.com

$$\operatorname*{minimize}_{x\in\mathbb{R}^n} \sum_{i\in I_1} \max\{0, 1-\langle x \mid a_i\rangle\} + \sum_{i\in I_2} \max\{0, 1+\langle x \mid a_i\rangle\} + \lambda\|x\|_1$$

Foe #2: Non-differentiability

A common paradigm:

1. Define an objective function
2. Optimize with an efficient algorithm, e.g., SGD with algorithmic differentiation (AD)

Issue: For many objective functions, a gradient does not exist.

Engineers*:

image: ripleys.com

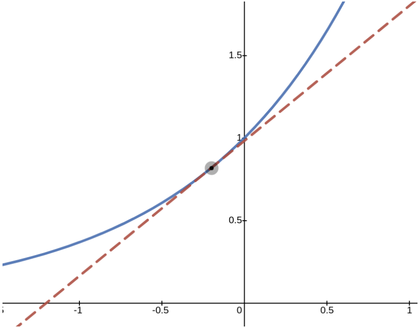
*:Some ~~mathematicians at heart~~ exceptions exist

How do we solve $\nabla f = 0$ when ∇f doesn't exist?

How do we solve $\nabla f = 0$ when ∇f doesn't exist?

If $f \in \Gamma_0(\mathcal{H})$ is differentiable at $x \in \mathcal{H}$, then

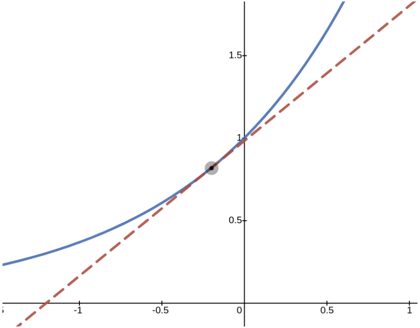
$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, \nabla f(x) \right\rangle + f(x) \leqslant f(y).$$



How do we solve $\nabla f = 0$ when ∇f doesn't exist?

A **subgradient** $g \in \mathcal{H}$ of $f : \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

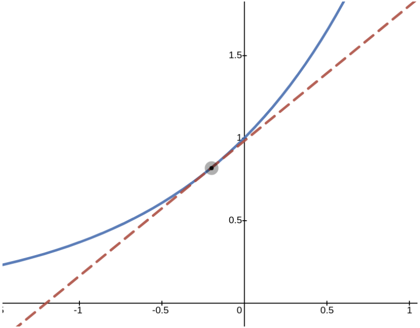


How do we solve $\nabla f = 0$ when ∇f doesn't exist?

A **subgradient** $g \in \mathcal{H}$ of $f: \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.



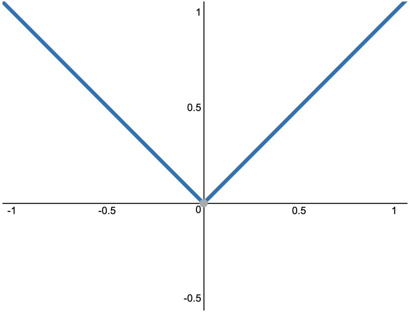
How do we solve $\nabla f = 0$ when ∇f doesn't exist?

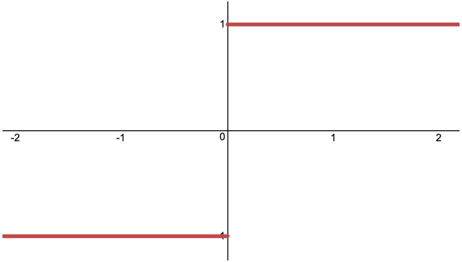
A **subgradient** $g \in \mathcal{H}$ of $f : \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.

Example: $f = |\cdot|$: What do we do at zero?





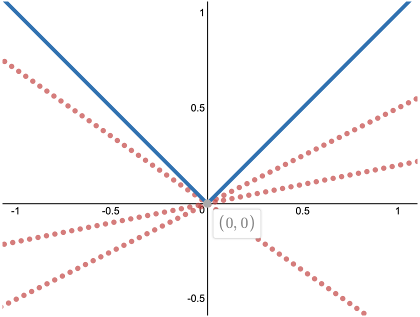
How do we solve $\nabla f = 0$ when ∇f doesn't exist?

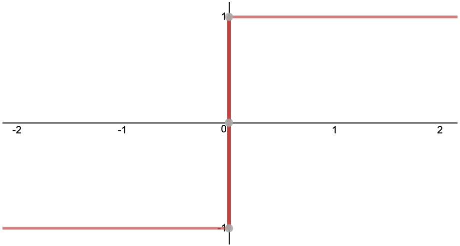
A **subgradient** $g \in \mathcal{H}$ of $f : \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.

Example: $f = |\cdot|$: What do we do at zero? $\partial f(0) = [-1, 1]$





How do we solve $\nabla f = 0$ when ∇f doesn't exist?

A **subgradient** $g \in \mathcal{H}$ of $f : \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.

Fermat's Rule

Let $x \in \mathcal{H}$. Then $x \in \operatorname{Argmin} f$ if and only if $0 \in \partial f(x)$.

How do we solve $\nabla f = 0$ when ∇f doesn't exist?

A **subgradient** $g \in \mathcal{H}$ of $f : \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \;\middle|\; g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.

Fermat's Rule

Let $x \in \mathcal{H}$. Then $x \in \operatorname{Argmin} f$ if and only if $0 \in \partial f(x)$.

Proof:

$$\begin{aligned} 0 \in \partial f(x) &\Leftrightarrow (\forall y \in \mathcal{H}) \quad \langle y - x | 0 \rangle + f(x) \leqslant f(y) \\ &\Leftrightarrow (\forall y \in \mathcal{H}) \quad f(x) \leqslant f(y) \\ &\Leftrightarrow x \in \operatorname{Argmin} \; f \end{aligned}$$

How do we solve $\nabla f = 0$ when ∇f doesn't exist?

A **subgradient** $g \in \mathcal{H}$ of $f \colon \mathcal{H} \to]-\infty, +\infty]$ at $x \in \mathcal{H}$ satisfies

$$(\forall y \in \mathcal{H}) \qquad \left\langle y - x \,\middle|\, g \right\rangle + f(x) \leqslant f(y).$$

The **subdifferential** $\partial f(x) \subset \mathcal{H}$ is the set containing all subgradients of f at x.

∂f is useful for developing both $\underbrace{\text{optimality criterion}}_{0 \in \partial f(x)}$ and algorithms.

Goal: "$0 \in \partial f(x)$". Which path do we take?

image: centralldm.es

Goal: "$0 \in \partial f(x)$". Which path do we take?

Some *provenly-convergent* (first-order) algorithm classes:

- Subgradient-projections (e.g., in [C. & ZW, *IEEE EUSIPCO*, 2020])
- Proximity operators (e.g., in [C., B., & ZW, *IEEE ICASSP*, 2022], [C. & ZW, *J. Approx. Theory*, 2021], [C. & ZW, *SIAM J. Imaging Sci.*, 2023])
- Conditional Gradient / "Frank-Wolfe" (e.g., in [ZW & P., 2024], [K., P., W., & ZW, *Opt. Methods. Softw.*, 2024])
- Abs-smooth Optimization (e.g., [K., P., W., & ZW, *Opt. Methods. Softw.*, 2024])
- Bundle methods, Barrier methods, Lagrangian methods, ...

Goal: "$0 \in \partial f(x)$". Which path do we take?

Some *provenly-convergent* (first-order) algorithm classes:

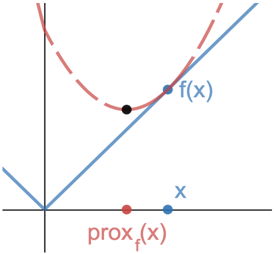
- Subgradient-projections (e.g., in [C. & ZW, *IEEE EUSIPCO*, 2020])
- **Proximity operators** (e.g., in [C., B., & ZW, *IEEE ICASSP*, 2022], [C. & ZW, *J. Approx. Theory*, 2021], [C. & ZW, *SIAM J. Imaging Sci.*, 2023])
- **Conditional Gradient / "Frank-Wolfe"** (e.g., in [ZW & P., 2024], [K., P., W., & ZW, *Opt. Methods. Softw.*, 2024])
- Abs-smooth Optimization (e.g., [K., P., W., & ZW, *Opt. Methods. Softw.*, 2024])
- Bundle methods, Barrier methods, Lagrangian methods, ...

Theoretically-sound optimization

Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\operatorname{prox}_f(x) = \underset{u \in \mathcal{H}}{\operatorname{Argmin}} \; f(u) + \frac{1}{2}\|x - u\|^2$$

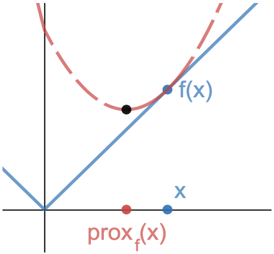


Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\mathrm{prox}_f(x) = \underset{u \in \mathcal{H}}{\mathrm{Argmin}} \; f(u) + \frac{1}{2}\|x - u\|^2$$

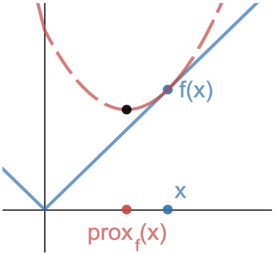
$\rightarrow$ For $f \in \Gamma_0(\mathcal{H})$ and $x \in \mathcal{H}$, $\mathrm{prox}_f(x)$ is unique.



Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\operatorname{prox}_f(x) = \underset{u \in \mathcal{H}}{\operatorname{Argmin}} \; f(u) + \frac{1}{2}\|x - u\|^2$$

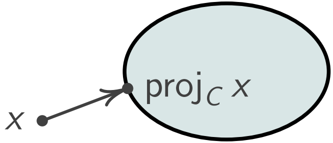


$\rightarrow$ For $f \in \Gamma_0(\mathcal{H})$ and $x \in \mathcal{H}$, $\operatorname{prox}_f(x)$ is unique. Defines an operator $\operatorname{prox}_f : \mathcal{H} \rightarrow \mathcal{H}$.

Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\mathrm{prox}_f(x) = \underset{u \in \mathcal{H}}{\mathrm{Argmin}} \ f(u) + \frac{1}{2}\|x - u\|^2$$



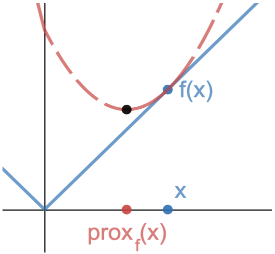
→ For $f \in \Gamma_0(\mathcal{H})$ and $x \in \mathcal{H}$, $\mathrm{prox}_f(x)$ is unique. Defines an operator $\mathrm{prox}_f : \mathcal{H} \to \mathcal{H}$.

→ Projection onto closed convex set C: $\mathrm{prox}_{\iota_C}(x) = \mathrm{Argmin}_{\,u \in C}\|x - u\|^2 = \mathrm{proj}_C\, x$.

Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\operatorname{prox}_f(x) = \underset{u \in \mathcal{H}}{\operatorname{Argmin}} \; f(u) + \frac{1}{2}\|x - u\|^2$$



→ For $f \in \Gamma_0(\mathcal{H})$ and $x \in \mathcal{H}$, $\operatorname{prox}_f(x)$ is unique. Defines an operator $\operatorname{prox}_f : \mathcal{H} \to \mathcal{H}$.

→ Projection onto closed convex set C: $\operatorname{prox}_{\iota_C}(x) = \operatorname{Argmin}_{u \in C}\|x - u\|^2 = \operatorname{proj}_C x$.

→ [Martinet, *Fr. Inf. Rech. Oper.*, 1970] (translated / modernized):
Let $f \in \Gamma_0(\mathcal{H})$ be such that $\operatorname{Argmin} f \neq \varnothing$.Let $\gamma > 0$, $x_0 \in \mathcal{H}$, and set

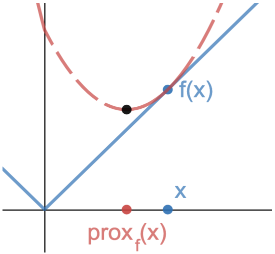
$$x_{n+1} = \operatorname{prox}_{\gamma f} x_n.$$

Then $(x_n)_{n \in \mathbb{N}} \rightharpoonup x^* \in \operatorname{Argmin} f$.

Proximity operators: a new hope

The **proximity operator of** f at $x \in \mathcal{H}$ is

$$\operatorname{prox}_f(x) = \underset{u \in \mathcal{H}}{\operatorname{Argmin}} \; f(u) + \frac{1}{2}\|x - u\|^2$$



→ For $f \in \Gamma_0(\mathcal{H})$ and $x \in \mathcal{H}$, $\operatorname{prox}_f(x)$ is unique. Defines an operator $\operatorname{prox}_f : \mathcal{H} \to \mathcal{H}$.

→ Projection onto closed convex set C: $\operatorname{prox}_{\iota_C}(x) = \operatorname{Argmin}_{u \in C} \|x - u\|^2 = \operatorname{proj}_C x$.

→ [Martinet, *Fr. Inf. Rech. Oper.*, 1970] (translated / modernized):
Let $f \in \Gamma_0(\mathcal{H})$ be such that $\operatorname{Argmin} f \neq \varnothing$.Let $\gamma > 0$, $x_0 \in \mathcal{H}$, and set

$$x_{n+1} = \operatorname{prox}_{\gamma f} x_n.$$

Then $(x_n)_{n \in \mathbb{N}} \rightharpoonup x^* \in \operatorname{Argmin} f$. Issue: prox_f might be hard to compute.

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

$(\text{prox}_{f_i})_{1 \leqslant i \leqslant m}$ are simpler than prox_f.

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

$(\mathrm{prox}_{f_i})_{1 \leqslant i \leqslant m}$ are simpler than prox_f.

Open-source repo's:

Python/Matlab:
`proximity-operator.net`,

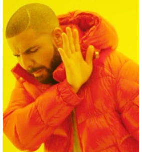
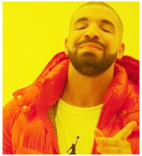
Julia:
`ProximalOperators.jl` (Github)

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

$(\mathrm{prox}_{f_i})_{1 \leqslant i \leqslant m}$ are simpler than prox_f.

Algorithmic “Splitting” mentality:

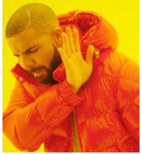
	prox_f
	$\mathrm{prox}_{f_1}, \ldots \mathrm{prox}_{f_m}$

Evolution of prox-based algorithms

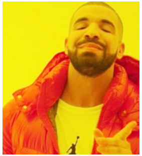
$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

$(\mathrm{prox}_{f_i})_{1 \leqslant i \leqslant m}$ are simpler than prox_f.

Algorithmic "Splitting" mentality:



prox_f



$\mathrm{prox}_{f_1}, \ldots \mathrm{prox}_{f_m}$

Pseudocode for many proximal splitting algorithms*

Require: Point $x_0 \in \mathcal{H}$, objective function f

for $n = 0, 1$ **to** $\ldots$ **do**
  \# Preprocess x_n
  for $i = 1$ **to** m **do**
    $y_{i,n+1} \leftarrow$ evaluation of $\mathrm{prox}_{f_i}(\cdot)$
  end for
  $x_{n+1} \leftarrow$ combine $(y_{i,n+1})_{1 \leqslant i \leqslant m}$
end for

*:e.g., Douglas-Rachford, ADMM, Chambolle-Pock, Forward-backward, Augmented Lagrangian, ...

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

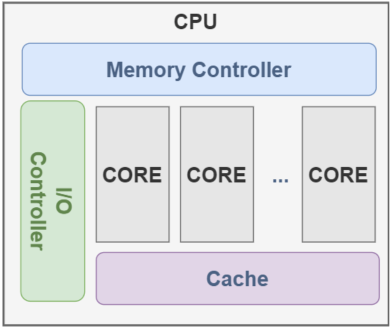


image: baeldung.com

Pseudocode for many proximal splitting algorithms*

Require: Point $x_0 \in \mathcal{H}$, objective function f

```
for n = 0, 1 to ... do
    # Preprocess x_n
    for i = 1 to m do
        y_{i,n+1} ← evaluation of prox_{f_i}(·)
    end for
    x_{n+1} ← combine (y_{i,n+1})_{1≤i≤m}
end for
```

*:e.g., Douglas-Rachford, ADMM, Chambolle-Pock, Forward-backward, Augmented Lagrangian, . . .

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

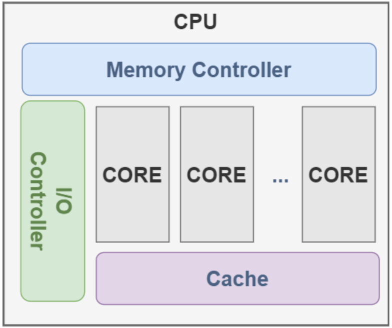


image: baeldung.com

Pseudocode for many proximal splitting algorithms*

Require: Point $x_0 \in \mathcal{H}$, objective function f

for $n = 0, 1$ **to** $\dots$ **do**
  # Preprocess x_n
  for $i = 1$ **to** m **do**
    $y_{i,n+1} \leftarrow$ evaluation of $\mathrm{prox}_{f_i}(\cdot)$
  end for
  $x_{n+1} \leftarrow$ combine $(y_{i,n+1})_{1 \leqslant i \leqslant m}$
end for

*:e.g., Douglas-Rachford, ADMM, Chambolle-Pock, Forward-backward, Augmented Lagrangian, . . .

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

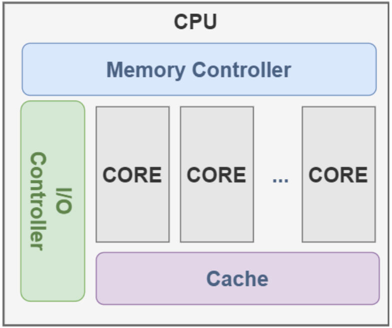


image: baeldung.com

Pseudocode for block-iterative proximal splitting algorithms

Require: Point $x_0 \in \mathcal{H}$, objective function f

for $n = 0, 1$ **to** $\ldots$ **do**

  \# Preprocess x_n; Select $I_n \subset \{1, \ldots, m\}$

  for $i = 1$ **to** m **do**

    if $i \in I_n$ **then**

      $y_{i,n+1} \leftarrow$ evaluation of $\operatorname{prox}_{f_i}(\cdot)$

    else

      $y_{i,n+1} \leftarrow y_{i,n}$

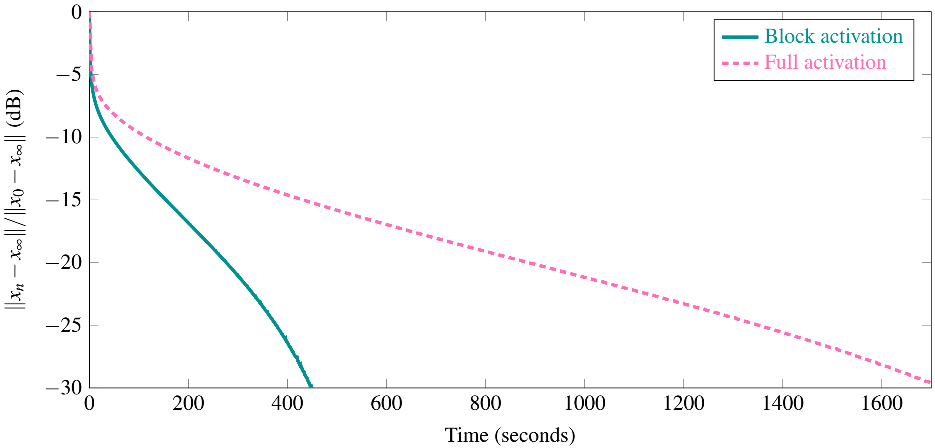
    end if

  end for

  $x_{n+1} \leftarrow$ combine $(y_{i,n+1})_{1 \leqslant i \leqslant m}$

end for

Block activation for image recovery ($m = 2$)



[C. & ZW, *SIAM J. Imaging Sci.*, 2022] Relative error for full-activation ($I_n = I$) versus block activation:

$$I_n = \begin{cases} \{1,2\}, & \text{if } n \equiv 0 \mod 5; \\ \{2\}, & \text{if } n \not\equiv 0 \mod 5. \end{cases}$$

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

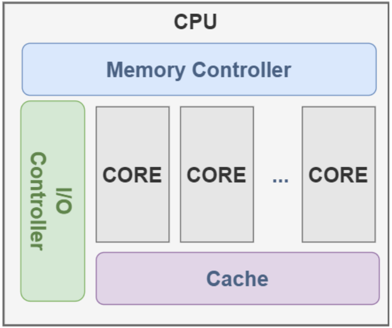


image: baeldung.com

Pseudocode for block-iterative proximal splitting algorithms*

Require: Point $x_0 \in \mathcal{H}$, objective function f

for $n = 0, 1$ **to** $\dots$ **do**
  # Preprocess x_n; Select $I_n \subset \{1, \dots, m\}$
  for $i = 1$ **to** m **do**
    if $i \in I_n$ **then**
      $y_{i,n+1} \leftarrow$ evaluation of $\mathrm{prox}_{f_i}(\cdot)$
    else
      $y_{i,n+1} \leftarrow y_{i,n}$
    end if
  end for
  $x_{n+1} \leftarrow$ combine $(y_{i,n+1})_{1 \leqslant i \leqslant m}$
end for

Evolution of prox-based algorithms

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

Two (currently separate) approaches:

→ Asynchronous updates

→ Extrapolated updates

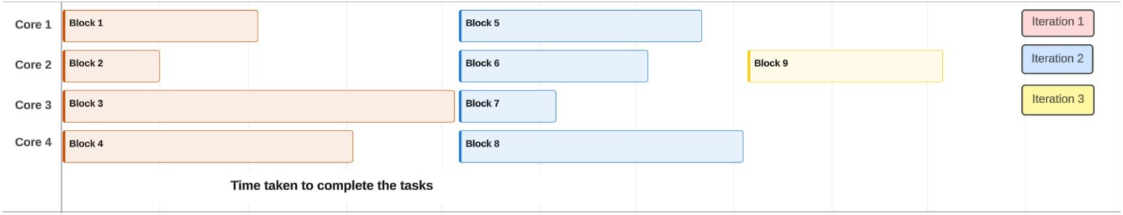
Pseudocode for block-iterative proximal splitting algorithms*

Require: Point $x_0 \in \mathcal{H}$, objective function f

for $n = 0, 1$ **to** $\ldots$ **do**
  $\#$ Preprocess x_n; Select $I_n \subset \{1, \ldots, m\}$
  for $i = 1$ **to** m **do**
    if $i \in I_n$ **then**
      $y_{i,n+1} \leftarrow$ evaluation of $\text{prox}_{f_i}(\cdot)$
    else
      $y_{i,n+1} \leftarrow y_{i,n}$
    end if
  end for
  $x_{n+1} \leftarrow$ combine $(y_{i,n+1})_{1 \leqslant i \leqslant m}$
end for

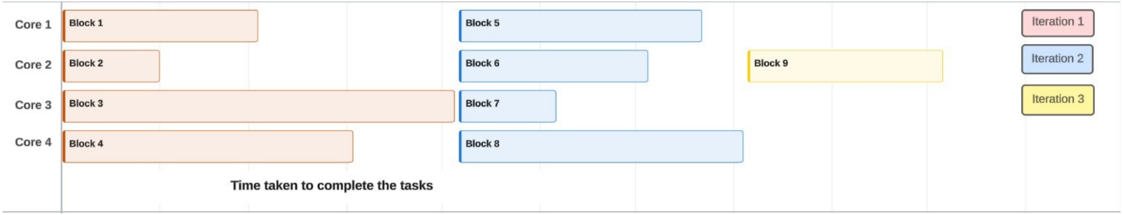
Evolution of prox-based algorithms

Parallel and Synchronous

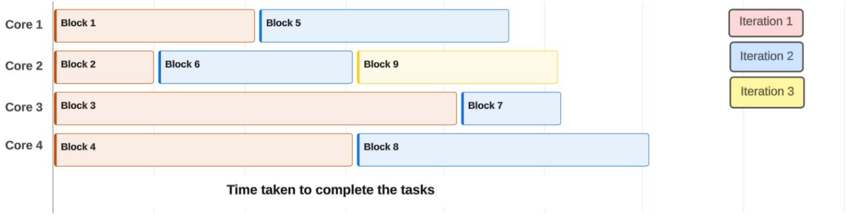


Evolution of prox-based algorithms

Parallel and Synchronous



Parallel and **Asynchronous**



Asynchrony: Projective Splitting Algorithms

[Eckstein & Svaiter, *Math Prog. A*, 2008]:

Coined "Projective splitting" (synchronous, not block-iterative).

[Combettes & Eckstein, *Math Prog. B*, 2018]:

Convergence proof with **asynchronous block-iterative** updates!

Asynchrony: Projective Splitting Algorithms

[Eckstein & Svaiter, *Math Prog. A*, 2008]:

Coined "Projective splitting" (synchronous, not block-iterative).
[Combettes & Eckstein, *Math Prog. B*, 2018]:

Convergence proof with **asynchronous block-iterative** updates!

[Combettes, Búi, & ZW, *IEEE ICASSP*, 2022]:

Numerical analysis (space and time complexity). For ML (training classifiers) and image processing, works better than other algorithms in its class.

Asynchrony: Projective Splitting Algorithms

[Eckstein & Svaiter, *Math Prog. A*, 2008]:

Coined "Projective splitting" (synchronous, not block-iterative).
[Combettes & Eckstein, *Math Prog. B*, 2018]:

Convergence proof with **asynchronous block-iterative** updates!

[Combettes, Búi, & ZW, *IEEE ICASSP*, 2022]:

Numerical analysis (space and time complexity). For ML (training classifiers) and image processing, works better than other algorithms in its class.
Only studied **synchronous** case!

Asynchrony: Projective Splitting Algorithms

[Eckstein & Svaiter, *Math Prog. A*, 2008]:

Coined "Projective splitting" (synchronous, not block-iterative).
[Combettes & Eckstein, *Math Prog. B*, 2018]:

Convergence proof with **asynchronous block-iterative** updates!

[Combettes, Búi, & ZW, *IEEE ICASSP*, 2022]:

Numerical analysis (space and time complexity). For ML (training classifiers) and image processing, works better than other algorithms in its class.
Only studied **synchronous** case!

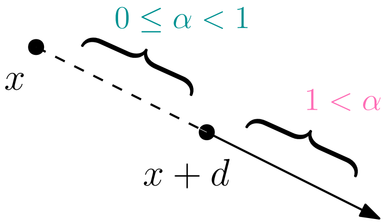
Dua, Goel, Sharma, & ZW (ongoing work):

Analysis and experimentation for **asynchronous** case. `AsyncProx.jl` in development.

Over-relaxation (extrapolation)

Given a "descent direction" d (e.g., combination of $(y_{i,n+1})_{1\leqslant i\leqslant m}$) from $x \in \mathcal{H}$,

$$x_+ = (1-\alpha)x + \alpha(x+d)$$



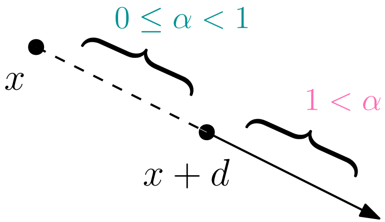
Over-relaxation (extrapolation)

Given a "descent direction" d (e.g., combination of $(y_{i,n+1})_{1\leqslant i\leqslant m}$) from $x \in \mathcal{H}$,

$$x_+ = (1-\alpha)x + \alpha(x+d)$$

$0 \leqslant \alpha < 1$: under-relaxation

$1 < \alpha$: over-relaxation



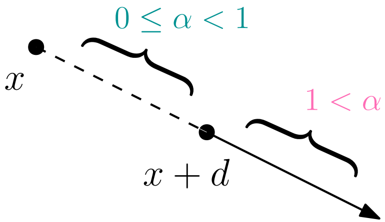
Over-relaxation (extrapolation)

Given a "descent direction" d (e.g., combination of $(y_{i,n+1})_{1\leqslant i\leqslant m}$) from $x \in \mathcal{H}$,

$$x_+ = (1-\alpha)x + \alpha(x+d)$$

$0 \leqslant \alpha < 1$: under-relaxation

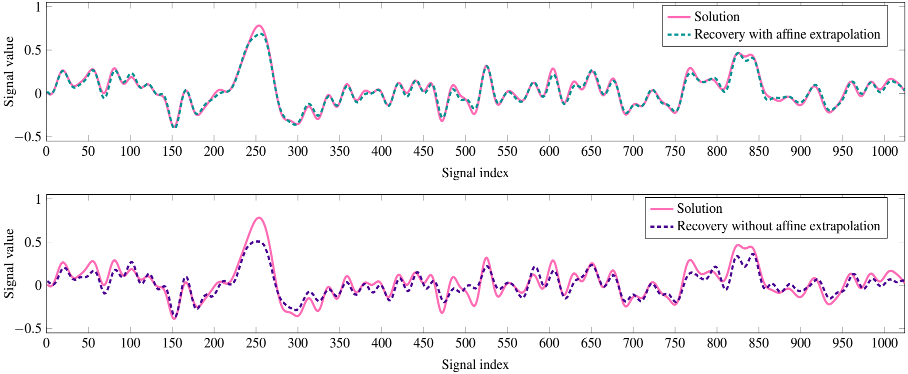
$1 < \alpha$: over-relaxation



$$x_{n+1} = x_n - \alpha_n \frac{1}{L}\nabla f(x_n), \qquad \text{where } \alpha_n \in [\varepsilon, 2-\varepsilon] \tag{GD}$$
$$= (1-\alpha_n)x_n + \alpha_n \left(x_n - \frac{1}{L}\nabla f(x_n)\right)$$

Over-relaxation for fixed-point problems

[Combettes & ZW, *J. Approx. Theory*, 2021]:
A strongly-convergent algorithm with affine-convex extrapolation.
Example: EEG data (minimal-norm solution to an ill-posed nonlinear inverse problem; recovery after 1000 iterations, < 1 minute.)



Theoretically-sound optimization

1. Motivation
2. Background: Theory vs practice
3. Proximity operators: Algorithmic bells and whistles
4. Splitting FW: What if the “usual” tools fail us?
5. More adventuring

They stole my horse!

Splitting problem setup

Given a smooth function $f:\mathbb{R}^n \to \mathbb{R}$ and compact convex sets $(C_i)_{1\leqslant i\leqslant m}$

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1\leqslant i\leqslant m} C_i, \tag{$\star$}$$

Applications: data science, matrix decomposition, quantum computing, combinatorial graph theory

They stole my horse!

Splitting problem setup

Given a smooth function $f : \mathbb{R}^n \to \mathbb{R}$ and compact convex sets $(C_i)_{1 \leqslant i \leqslant m}$

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \tag{$\star$}$$

Applications: data science, matrix decomposition, quantum computing, combinatorial graph theory

What if $(\mathrm{proj}_{C_i})_{1 \leqslant i \leqslant m}$ are too expensive? $\leftarrow$ e.g., in high-dimensional settings!

A spark of inspiration

Frank-Wolfe / "Conditional gradient" alg. [Naval Res. Logist. Quart., 1956]

Given a smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$ and a nonempty compact convex set C,

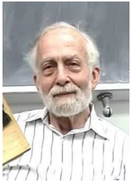
$$\text{minimize} \;\; f(x) \;\; \text{subject to} \;\; x \in C$$

Instead of projecting, use a *linear minimization oracle* of C,

$$\mathrm{LMO}_C\colon y \mapsto p \in \operatorname{Argmin}_{x\in C} \langle y \mid x \rangle \tag{LMO}$$

$$x_{n+1} = x_n + \frac{1}{n+1}\Big(\mathrm{LMO}_C(\nabla f(x_n)) - x_n\Big)$$

Marguerite Frank



Philip Wolfe

efficiency(LMOs) == efficiency(lembus bread)

[Combettes/Pokutta, '21]: For many $\underbrace{\text{constraints}}$, C, proj_C is **more expensive** than LMO_C.
(e.g., nuclear norm ball, ℓ_1 ball, probability simplex, Birkhoff polytope, general LP, ...)

efficiency(LMOs) == efficiency(lembus bread)

[Combettes/Pokutta, '21]: For many $\underbrace{\text{constraints}}$, C, proj_C is **more expensive** than LMO_C.
(e.g., nuclear norm ball, ℓ_1 ball, probability simplex, Birkhoff polytope, general LP, ...)

Example: Nuclear norm ball

For $x \in \mathbb{R}^{n\times n}$

$$\|x\|_{nuc} = \sum_{1\leqslant i\leqslant n} \sigma_i(x).$$

For $\beta \geqslant 0$, $C = \{x \in \mathbb{R}^{n\times n} \mid \|x\|_{nuc} \leqslant \beta\}$,

`efficiency`(**LMOs**) `==` `efficiency`(**lembus bread**)

[Combettes/Pokutta, '21]: For many $\underbrace{\text{constraints}}$, C, proj_C is **more expensive** than LMO_C.
(e.g., nuclear norm ball, ℓ_1 ball, probability simplex, Birkhoff polytope, general LP, ...)

Example: Nuclear norm ball

For $x \in \mathbb{R}^{n \times n}$

$$\|x\|_{nuc} = \sum_{1 \leqslant i \leqslant n} \sigma_i(x).$$

For $\beta \geqslant 0$, $C = \{x \in \mathbb{R}^{n \times n} \mid \|x\|_{nuc} \leqslant \beta\}$,

$\text{proj}_C(x)$: requires a full SVD!
$(\sigma_1, \ldots, \sigma_n, U, V)$, where $x = U \text{diag}(\sigma_1, \ldots, \sigma_n) V^\top$

Full SVD

$n = 500$: `0.11 sec`

$n = 1000$: `0.47 sec`

$n = 2000$: `4.87 sec`

`efficiency(`**LMOs**`) == efficiency(`**lembus bread**`)`

[Combettes/Pokutta, '21]: For many $\underbrace{\text{constraints}}$, C, proj_C is **more expensive** than LMO_C.
(e.g., nuclear norm ball, ℓ_1 ball, probability simplex, Birkhoff polytope, general LP, ...)

Example: Nuclear norm ball

For $x \in \mathbb{R}^{n\times n}$

$$\|x\|_{nuc} = \sum_{1\leqslant i\leqslant n} \sigma_i(x).$$

For $\beta \geqslant 0$, $C = \{x \in \mathbb{R}^{n\times n} \mid \|x\|_{nuc} \leqslant \beta\}$,

$\text{proj}_C(x)$: requires a full SVD!
$(\sigma_1, \ldots, \sigma_n, U, V)$, where $x = U\text{diag}(\sigma_1, \ldots, \sigma_n)V^\top$

$\text{LMO}_C(x)$: requires only first singular value/vectors
$(\sigma_1, U_1, V_1^\top)$

Full SVD

$n = 500$: `0.11 sec`

$n = 1000$: `0.47 sec`

$n = 2000$: `4.87 sec`

Just $(\sigma_1, U_1, V_1^\top)$

$n = 500$: `0.0081 sec`

$n = 1000$: `0.056 sec`

$n = 2000$: `0.638 sec`

Sure would be nice if we could use splitting...

$$\text{minimize } f(x) \text{ subject to } x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \tag{$\star$}$$

Sure would be nice if we could use splitting. . .

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \qquad (\star)$$

Issue: Computing the LMO for $\bigcap_{1 \leqslant i \leqslant m} C_i$ is prohibitively expensive.

Sure would be nice if we could use splitting. . .

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \tag{$\star$}$$

Issue: Computing the LMO for $\bigcap_{1\leqslant i\leqslant m} C_i$ is prohibitively expensive.

$(\mathrm{LMO}_{C_i})_{1\leqslant i\leqslant m}$ are easier to evaluate (e.g., repository: `FrankWolfe.jl`)

Sure would be nice if we could use splitting. . .

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \tag{$\star$}$$

Issue: Computing the LMO for $\bigcap_{1\leqslant i\leqslant m} C_i$ is prohibitively expensive.

$(\mathrm{LMO}_{C_i})_{1\leqslant i\leqslant m}$ are easier to evaluate (e.g., repository: `FrankWolfe.jl`)

LMO-based *splitting algorithms*, enforce constraints via LMOs for the individual sets

$$\text{Use}\ \mathrm{LMO}_{C_1},\ \ \mathrm{LMO}_{C_2},\ \ \ldots\ \ \text{instead of}\ \ \mathrm{LMO}_{\left(\bigcap_{1\leqslant i\leqslant m} C_i\right)}$$

Sure would be nice if we could use splitting...

$$\text{minimize } f(x) \text{ subject to } x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \qquad (\star)$$

Issue: Computing the LMO for $\bigcap_{1 \leqslant i \leqslant m} C_i$ is prohibitively expensive.

$(\mathrm{LMO}_{C_i})_{1 \leqslant i \leqslant m}$ are easier to evaluate (e.g., repository: `FrankWolfe.jl`)

LMO-based *splitting algorithms*, enforce constraints via LMOs for the individual sets

Use LMO_{C_1}, LMO_{C_2}, ... instead of $\mathrm{LMO}_{(\bigcap_{1 \leqslant i \leqslant m} C_i)}$

Relatively little has been done in this field.

Sure would be nice if we could use splitting...

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \qquad (\star)$$

Issue: Computing the LMO for $\bigcap_{1\leqslant i\leqslant m} C_i$ is prohibitively expensive.

$(\mathrm{LMO}_{C_i})_{1\leqslant i\leqslant m}$ are easier to evaluate (e.g., repository: `FrankWolfe.jl`)

LMO-based *splitting algorithms*, enforce constraints via LMOs for the individual sets

Use $\mathrm{LMO}_{C_1}, \ \mathrm{LMO}_{C_2}, \ \ldots$ instead of $\mathrm{LMO}_{(\bigcap_{1\leqslant i\leqslant m} C_i)}$

Relatively little has been done in this field.

$\rightarrow$ Unlike projections, LMOs are discontinuous.

Sure would be nice if we could use splitting. . .

$$\text{minimize } f(x) \text{ subject to } x \in \bigcap_{1 \leqslant i \leqslant m} C_i, \qquad (\star)$$

Issue: Computing the LMO for $\bigcap_{1 \leqslant i \leqslant m} C_i$ is prohibitively expensive.

$(\mathrm{LMO}_{C_i})_{1 \leqslant i \leqslant m}$ are easier to evaluate (e.g., repository: `FrankWolfe.jl`)

LMO-based *splitting algorithms*, enforce constraints via LMOs for the individual sets

$$\boxed{\text{Use } \mathrm{LMO}_{C_1},\ \mathrm{LMO}_{C_2},\ \ldots \ \text{instead of}\ \mathrm{LMO}_{\left(\bigcap_{1 \leqslant i \leqslant m} C_i\right)}}$$

Relatively little has been done in this field.

$\rightarrow$ Unlike projections, LMOs are discontinuous.

$\rightarrow$ "State-of-the-art" relies on inexact prox-based algorithms.

It's aliiive!

[ZW & P, 2024]: Split Conditional Gradient Algorithm

Require: Point $x_0 \in \frac{1}{m}\sum_{1\leqslant i\leqslant m} C_i$, smooth function f

```
for t = 0, 1 to ... do
    Choose penalty parameter λ_t ∈ ]0, +∞[
    Choose step size γ_t ∈ ]0, 1]
    g_t ← ∇f(x_t)
    for i = 1 to m do
        v_t^i ← LMO_{C_i}(g_t + λ_t(x_t^i − x_t))
        x_{t+1}^i ← x_t^i + γ_t(v_t^i − x_t^i)
    end for
    x_{t+1} ← (1/m) Σ_{1⩽i⩽m} x_{t+1}^i
end for
```

$$\begin{aligned}
&\textbf{for } t = 0, 1 \textbf{ to } \dots \textbf{ do}\\
&\quad \text{Choose penalty parameter } \lambda_t \in \,]0, +\infty[\\
&\quad \text{Choose step size } \gamma_t \in \,]0, 1]\\
&\quad g_t \leftarrow \nabla f(x_t)\\
&\quad \textbf{for } i = 1 \textbf{ to } m \textbf{ do}\\
&\quad\quad \boldsymbol{v}_t^i \leftarrow \mathrm{LMO}_{C_i}(g_t + \lambda_t(\boldsymbol{x}_t^i - x_t))\\
&\quad\quad \boldsymbol{x}_{t+1}^i \leftarrow \boldsymbol{x}_t^i + \gamma_t(\boldsymbol{v}_t^i - \boldsymbol{x}_t^i)\\
&\quad \textbf{end for}\\
&\quad x_{t+1} \leftarrow \frac{1}{m}\sum_{1\leqslant i\leqslant m} \boldsymbol{x}_{t+1}^i\\
&\textbf{end for}
\end{aligned}$$

Practical advantages:

→ Uses individual LMOs

→ Lowest-known # LMO calls: one LMO per set (per iteration)

It's aliiive!

[ZW & P, 2024]: Split Conditional Gradient Algorithm

Require: Point $x_0 \in \frac{1}{m}\sum_{1\leqslant i\leqslant m} C_i$, smooth function f

```
for t = 0, 1 to ... do
    Choose penalty parameter λ_t ∈ ]0, +∞[
    Choose step size γ_t ∈ ]0, 1]
    g_t ← ∇f(x_t)
    for i = 1 to m do
        v_t^i ← LMO_{C_i}(g_t + λ_t(x_t^i − x_t))
        x_{t+1}^i ← x_t^i + γ_t(v_t^i − x_t^i)
    end for
    x_{t+1} ← (1/m) Σ_{1⩽i⩽m} x_{t+1}^i
end for
```

Practical advantages:

$\rightarrow$ Uses individual LMOs

$\rightarrow$ Lowest-known # LMO calls: one LMO per set (per iteration)

Q: Does it actually solve $(\star)$?

A: Yes.

$\gamma_t = \mathcal{O}(1/\sqrt{t})$ and
$\lambda_t = \mathcal{O}(\ln t)$ work.
(whether or not f is convex).

Convergence

Theorem ([ZW & P., 2024])

Let f be L-smooth and let $(C_i)_{1\leqslant i\leqslant m}$ be nonempty compact convex subsets of $\mathcal{H}$ such that $\bigcap_{1\leqslant i\leqslant m} C_i \neq \varnothing$. Let $\lambda_0 > 0$ and $\lambda_{t+1} = \lambda_t + (\sqrt{t}+2)^{-2}$ and $\gamma_t = 2/(\sqrt{t}+2)$. If f is **convex**, then

$$f\left(\frac{1}{m}\sum_{1\leqslant i\leqslant m} \boldsymbol{x}_t^i\right) \to \inf_{x\in\bigcap_{1\leqslant i\leqslant m} C_i} f(x)$$

and every accumulation point $\boldsymbol{x}_\infty$ of $(\boldsymbol{x}_t)_{t\in\mathbb{N}}$ produces a solution $\frac{1}{m}\sum_{1\leqslant i\leqslant m} \boldsymbol{x}_\infty^i \in \bigcap_{1\leqslant i\leqslant m} C_i$ such that $f(\frac{1}{m}\sum_{1\leqslant i\leqslant m} \boldsymbol{x}_\infty) = \inf_{x\in\bigcap_{1\leqslant i\leqslant m} C_i} f(x)$.

Nonconvex convergence results too: arXiv:2311.05381

Theoretically-sound optimization

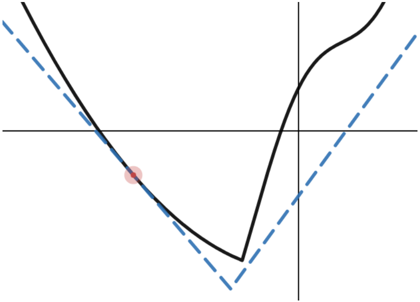
Abs-smooth optimization

Abs-smooth functions f include compositions of smooth functions, max, min, and $|\cdot|$. ...

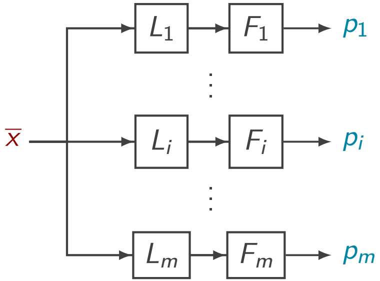
→ Loss functions for multilayer Neural Networks with smooth and/or ReLU activation.

→ Allows one to find a local minimizer on non-convex objective functions!

→ Future work: Improve scalability.



Nonlinear inverse problems

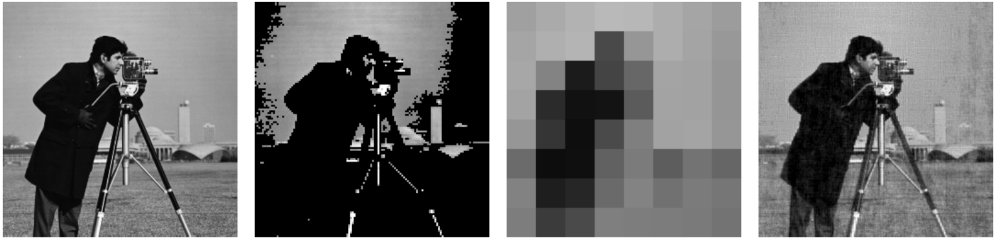


Given $(p_i)_{1 \leqslant i \leqslant m}$, find $x^* \in \mathcal{H}$ such that

$$(\forall i \in \{1, \dots, m\}) \quad F_i(L_i x^*) = p_i$$

L_i are bounded linear operators, and
$F_i \approx$ proximity operators.

$\rightarrow$ To-do: Stability analysis



Outlook and future work

- Improved convergence rates and acceleration
- Block-iterative Frank-Wolfe algorithms
- Efficient prox/ LMO algorithms
- ...

Outlook and future work

- Improved convergence rates and acceleration
- Block-iterative Frank-Wolfe algorithms
- Efficient prox/ LMO algorithms
- . . .

Potential collaborators: Hala Nelson, Minah Oh, Roger Thelwell, and more!

For students:

Proofs (MATH 245), sequences and series (236), gradients (237), linear algebra (238/300/434), optimization theory (340), coding experience (248/250/448/449), analysis (410/411).

REUs & Grad School in Berlin: `iol.zib.de` (Opt/ML), `math-berlin.de`

Thank you for your attention!

References

M. N. Búi, P. L. Combettes and ZW, Block-activated algorithms for multicomponent fully nonsmooth minimization
Proc. IEEE Int. Conf. Acoust. Speech Signal Process., pp 5428–5432, 2022.

P. L. Combettes, A. M. McDonald, C. A. Micchelli, and M. Pontil, Learning with optimal interpolation norms
Numer. Algorithms, vol. 81, no. 2, pp. 695-–717, 2019.

P. L. Combettes and ZW, Signal recovery from inconsistent nonlinear observations
Proc. IEEE Int. Conf. Acoust. Speech Signal Process., pp 5872—5876, 2022.

P. L. Combettes and ZW, A fixed point framework for recovering signals from nonlinear transformations
Proc. IEEE Eur. Signal Proc., pp 2120-–2124, 2020.

P. L. Combettes and ZW, Reconstruction of functions from prescribed proximal points
J. Approx. Theory, vol. 268, no. 105606, 2021

References

P. L. Combettes and ZW, A variational inequality model for the construction of signals from inconsistent nonlinear equations
SIAM J. Imaging Sci., vol. 15, no. 1, pp. 84–109, 2022

C. Combettes and S. Pokutta, Complexity of linear minimization and projection on some sets
Oper. Res. Lett., vol. 49, no. 4, pp. 565–571, 2021

M. Frank and P. Wolfe, An algorithm for quadratic programming
Naval Res. Logist. Quart., vol. 3, iss. 1–2, pp. 95–110, 1956

T. Kreimeier, S. Pokutta, A. Walther, and ZW, On a Frank-Wolfe approach for abs-smooth functions
Opt. Methods and Softw., DOI: 10.1080/10556788.2023.2296985, 2024

B. Martinet, régularisation d'inéquations variationnelles par approximations successives
Fr. Inf. Rech. Oper., Série rouge, 4 (R3):154–158, 1970.

References

C. S. Sartori and L. S. Buriol, A study on the pickup and delivery problem with time windows: Matheuristics and new instances
Comput. Oper. Res., vol. 124, art. 105065, 2020.

N. Torelli, D. Papp, and J. Unkelbach, Spatiotemporal fractionation schemes for stereotactic radiosurgery of multiple brain metastases
Med. Phys., vol. 50, no. 8, pp. 5095-5114, 2023.

ZW and S. Pokutta, Splitting the conditional gradient algorithm
arXiv:2311.05381, 2024

Supplement

$$f = \sum_{1 \leqslant i \leqslant m} f_i$$

$(\mathrm{prox}_{f_i})_{1 \leqslant i \leqslant m}$ are simpler than prox_f.

e.g., $\boxed{\text{Find } x \in \mathbb{S}^n_+ \cap [\alpha, \beta]^{N \times N}}$

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \iota_{\mathbb{S}^n_+}(x) + \iota_{[\alpha,\beta]^N}(x)$$

prox_f is intractable.

$$\left.\begin{aligned} \mathrm{prox}_{f_1} &= \mathrm{proj}_{[\alpha,\beta]^N} \\ \mathrm{prox}_{f_2} &= \mathrm{proj}_{\mathbb{S}^n_+} \end{aligned}\right\} \text{known in closed-form.}$$