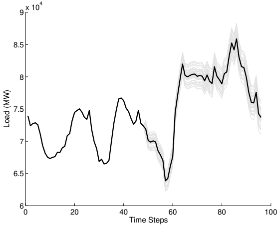
Vers des décisions plus résiliantes:
applications de l'optimisation non-lisse
en production électrique et en machine learning

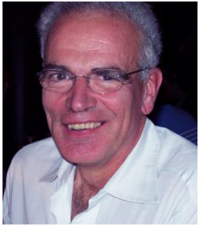
Jérôme MALICK

EDF Dec. 2025

Teasing...



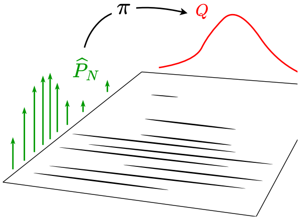
load scenarios



C. Lemaréchal

histogram reshaping

renewable energy



transport optimal

flying pigs

Nonsmooth objective functions are everywhere...

$$\min_{x \in C} F(x)$$

Max functions

$$F(x) = \sup_{u \in U} \; h(u, x)$$

- robust optimization, stochastic optimization, decomposition methods
- relaxations of combinatorial problems

Nonsmooth regularization

$$F(x) = f(x) + g(x)$$

- image/signal processing, inverse problems
- sparsity-inducing regularizers in machine learning

Nonsmooth composition

$$F(x) = g \circ c(x)$$

- risk-averse optimization, eigenvalue optimization
- deep learning: nonsmooth activation, implicit layers

Probability functions

$$F(x) = \mathbb{P}\big(h(x, \xi) \leqslant 0\big)$$

- optimization under uncertainty, energy optimization

So what ?...

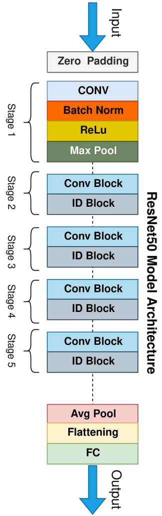
Is nonsmoothness really important ? useful ?

Why not just ignoring it ?

- **Ex:** nonsmoothness in deep learning
  (with RELU, max-pooling or implicit layers)
- Just apply SGD with back-prog

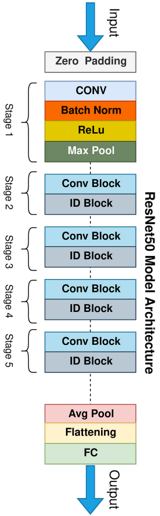
Why not smoothing it ?

- Smoothing by (inf-)convolution (e.g. Moreau regularization)
- Smoothings by overparameterization, ad hoc, or...



So what ?...

Is nonsmoothness really important ? useful ?



Why not just ignoring it ?

- **Ex:** nonsmoothness in deep learning
  (with RELU, max-pooling or implicit layers)
- Just apply SGD with back-prog

Why not smoothing it ?

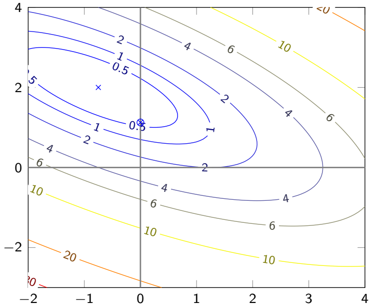
- Smoothing by (inf-)convolution (e.g. Moreau regularization)
- Smoothings by overparameterization, ad hoc, or...

My point: nonsmoothness is (nice and) relevant !

Example: ℓ_1-regularized least-squares (1/2)

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2}\|Ax-y\|^2 + \lambda\|x\|_1 \qquad \text{(LASSO)}$$

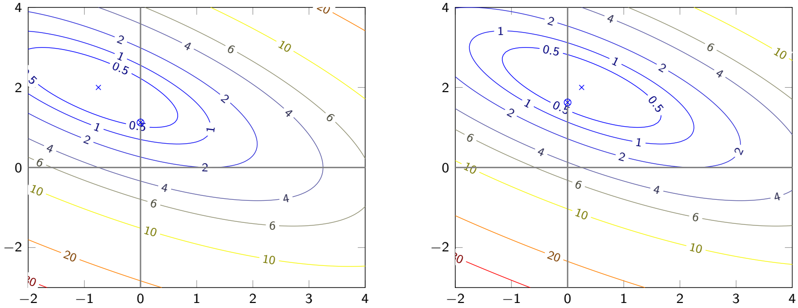
Illustration (on an instance with $d = 2$)



Example: ℓ_1-regularized least-squares (1/2)

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2}\|Ax - y\|^2 \; + \; \lambda\|x\|_1 \qquad \text{(LASSO)}$$

Illustration (on an instance with $d = 2$)



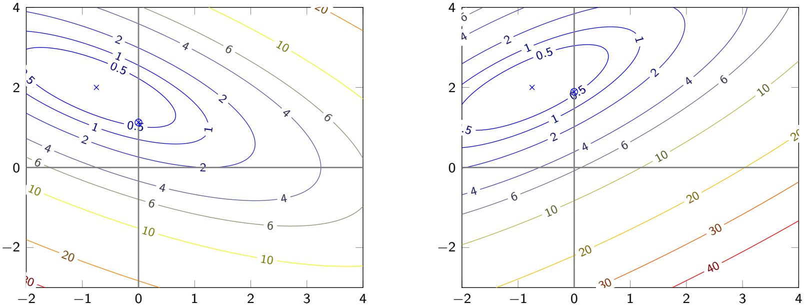
the support of optimal solutions is stable under small perturbations

Nonsmoothness traps solutions in low-dimensional manifolds

Example: ℓ_1-regularized least-squares (1/2)

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2}\|Ax - y\|^2 \;+\; \lambda\|x\|_1 \qquad \text{(LASSO)}$$

Illustration (on an instance with $d = 2$)



the support of optimal solutions is stable under small perturbations

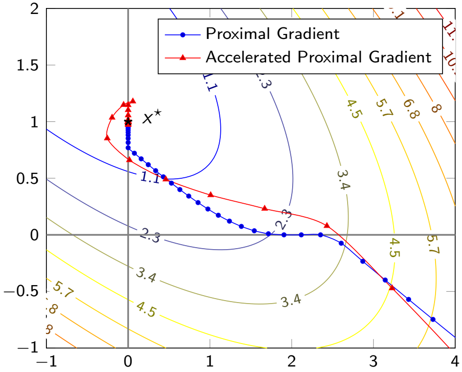
Nonsmoothness traps solutions in low-dimensional manifolds

Example: ℓ_1-regularized least-squares (2/2)

$$\min_{x \in \mathbb{R}^d} \quad \frac{1}{2}\|Ax - y\|^2 \; + \; \lambda\|x\|_1 \qquad \text{(LASSO)}$$

Example: ℓ_1-regularized least-squares (2/2)

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{2}\|Ax-y\|^2 \; + \; \lambda\|x\|_1 \qquad \text{(LASSO)}$$



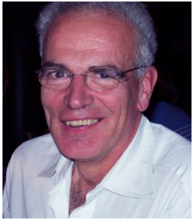
(proximal-gradient) algorithms produce iterates...

...that eventually have the same support as the optimal solution

Nonsmoothness attracts (proximal) algorithms

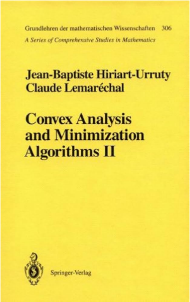
Remark: smooth but stiff problems

J.-B. Hiriart-Urruty



C. Lemaréchal

"There is no clear cut between functions that are smooth and functions that are not.
In-between there is a rather fuzzy boundary of stiff functions"



In sharp contrast with smoothing-like approaches:

- Toy example from the book (Section VIII.3.3): for a smooth problem, run usual algorithms
  nonsmooth methods (prox/level-bundle) $\gg$ smooth methods (gradient, conj. grad., q-Newton)
- Real-life example in energy optimization :
  - problem of managment of reservoirs : smooth
  - state-of-the-art algos to solve it : nonsmooth

Nonsmoothness can help, even for (difficult) smooth problems

Today's message

Nonsmoothness is sometimes useful, sometimes unavoidable – and always nice-looking

(Modest) Goals of this talk:

- Advocacy for nonsmooth optimization
- Spotlights on 2 applications
- #1: in electricity generation, handling uncertainty
- #2: in machine learning, towards resilience and fairness
- High level: underline ideas, duality, models...

No theorems ! No algorithms ! (Almost) No references !

Spotlight #1: handling uncertainty

Nonsmooth optimization

for electricity generation management

Optimization of electricity generation

In France: electricity is (essentially) generated by N EDF units

nuclear 63%

renewables 14%

oil/gaz/coal 12%

hydro 17%

Question : finding "optimal" daily production schedules

Optimization of electricity generation

In France: electricity is (essentially) generated by N EDF units

nuclear 63%

renewables 14%

oil/gaz/coal 12%

hydro 17%

Question : finding "optimal" daily production schedules

Day-to-day optimization of production ("unit-commitment")

$$\left(\begin{array}{l}\text{simplified}\\ \text{model}\end{array}\right) \quad \begin{cases} \min \quad \sum_i c_i^\top x_i & \text{(production costs)} \\ \quad \sum_i x_i = d & \text{(demand constraints)} \\ \quad (x_1, \ldots, x_N) \in X_1 \times \cdots \times X_N & \text{(operational constraints)} \end{cases}$$

Hard optimization problem: large-scale, heterogeneous, complex ($\geqslant 10^6$ variables, $\geqslant 10^6$ constraints)

Out of reach for (mixed-integer linear) solvers... But where is the nonsmoothness ?

Optimization of electricity generation

In France: electricity is (essentially) generated by N EDF units

nuclear 63%

renewables 14%

oil/gaz/coal 12%

hydro 17%

Question : finding "optimal" daily production schedules

Day-to-day optimization of production ("unit-commitment")

$$\begin{pmatrix}\text{simplified}\\ \text{model}\end{pmatrix} \quad \begin{cases} \min \quad \sum_i c_i^\top x_i & \text{(production costs)} \\ \quad \sum_i x_i = d \quad \leftarrow u \in \mathbb{R}^T & \text{(demand constraints)} \\ \quad (x_1, \ldots, x_N) \in X_1 \times \cdots \times X_N & \text{(operational constraints)} \end{cases}$$

Hard optimization problem: large-scale, heterogeneous, complex ($\geqslant 10^6$ variables, $\geqslant 10^6$ constraints)

Out of reach for (mixed-integer linear) solvers... But where is the nonsmoothness ?

Solution: duality, decomposition, and nonsmoothness

- Dual function (concave)

$$\theta(u) = \left\{ \begin{array}{ll} \min & \displaystyle\sum_{i=1}^{N} c_i^\top x_i + \sum_{t=1}^{T} u^t \Big(d^t - \sum_{i=1}^{N} x_i^t \Big) \\ & (x_1, \dots, x_N) \in X_1 \times \cdots \times X_N \end{array} \right.$$

Solution: duality, decomposition, and nonsmoothness

- Dual function (concave)

$$\theta(u) = \begin{cases} \min & \sum_{i=1}^{N} {c_i}^\top x_i + \sum_{t=1}^{T} u^t \left(d^t - \sum_{i=1}^{N} x_i^t\right) \\ & (x_1, \dots, x_N) \in X_1 \times \cdots \times X_N \end{cases}$$

- Dualizing the coupling constraint makes it decomposable by units

$$\theta(u) = d^\top u + \sum_{i=1}^{N} \theta_i(u)$$

$$\theta_i(u) = \begin{cases} \min & (c_i - u)^\top x_i \\ & x_i \in X_i \end{cases}$$

Solution: duality, decomposition, and nonsmoothness

- Dual function (concave)

$$\theta(u) = \left\{ \begin{array}{ll} \min & \displaystyle\sum_{i=1}^{N} c_i^\top x_i + \sum_{t=1}^{T} u^t \Big(d^t - \sum_{i=1}^{N} x_i^t \Big) \\ & (x_1, \dots, x_N) \in X_1 \times \cdots \times X_N \end{array} \right.$$

- Dualizing the coupling constraint makes it decomposable by units

$$\theta(u) = d^\top u + \sum_{i=1}^{N} \theta_i(u)$$

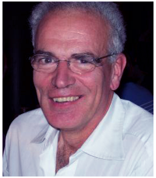
$$\theta_i(u) = \left\{ \begin{array}{ll} \min & (c_i - u)^\top x_i \\ & x_i \in X_i \end{array} \right.$$

- Nonsmooth algorithm:
  inexact prox. bundle [Lemaréchal '75... '95]

Nonsmooth optimization algorithm

shadow prices ↓ ↑ decentralized productions




C. Lemarechal


S. Charousset


A. Renaud

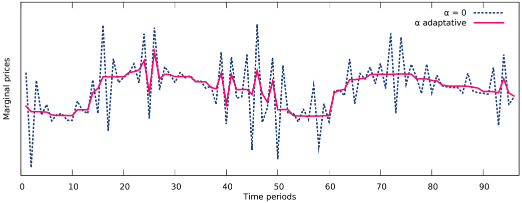
- Research in the 1990's
- In action in early 2000's
- Idea still rules in 2020's

  talk of Sandrine at SMAI-MODE 2024

You know all this better than me...

Modest contributions on some algorithmic aspects

- Acceleration of the bundle method (using coarse linearizations) [Malick, Oliveira, Zaourar '15]
- (Level) asynchronous bundle algorithm [Iutzeler, Malick, Oliveira '18]
- Denoising dual solutions (by TV-regularization) [Zaourar, Malick '13]
- Introducing weather uncertainty in the model
  - robust version of the problem + bundle method [van Ackooij, Lebbe, Malick '16]
  - 2-stage stochastic version + double decomposition algorithm [van Ackooij, Malick '15]

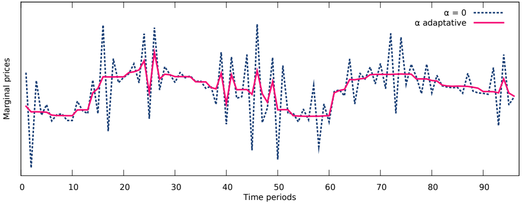


You know all this better than me...

Modest contributions on some algorithmic aspects

- Acceleration of the bundle method (using coarse linearizations) [Malick, Oliveira, Zaourar '15]
- (Level) asynchronous bundle algorithm [Iutzeler, Malick, Oliveira '18]
- Denoising dual solutions (by TV-regularization) [Zaourar, Malick '13]
- Introducing weather uncertainty in the model
  - robust version of the problem + bundle method [van Ackooij, Lebbe, Malick '16]
  - 2-stage stochastic version + double decomposition algorithm [van Ackooij, Malick '15]

...handling uncertainty adds extra nonsmoothness ☺

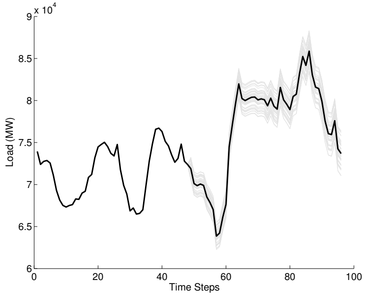


Two-stage stochastic unit-commitment

- The schedule x is sent to the grid-operator before being activated

- At certain moments in time the production schedule can be updated

- At time τ, we have the observed load $\xi_1, \ldots, \xi_\tau$ and the current best forecast $\xi_{\tau+1}, \ldots, \xi_T$



- [van Ackooij, Malick '15] proposes a stochastic 2-stage problem:

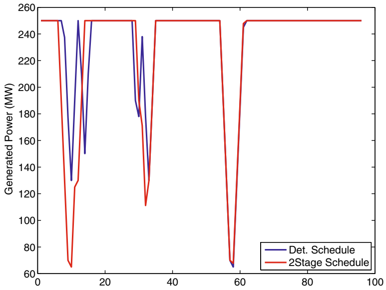
$$\left\{ \begin{array}{ll} \min & c^\top x + \mathbb{E}[c(x,\xi)] \\ & x \in X, \quad \sum_i x_i = d \end{array} \right. \qquad \text{where } c(x,\xi) = \left\{ \begin{array}{ll} \min & c^\top y \\ & y \in X, \quad \sum_i y_i = \xi \\ & y \text{ coincides with } x \text{ on } 1, \ldots, \tau \end{array} \right.$$

  - the first and second stage are full unit-commitment problems
  - 2nd stage model: same as 1st stage but with smaller horizon
  - fine operational modeling vs difficult to compute
  - complexity of $c(x,\xi)$ only allows for simple modeling of randomness

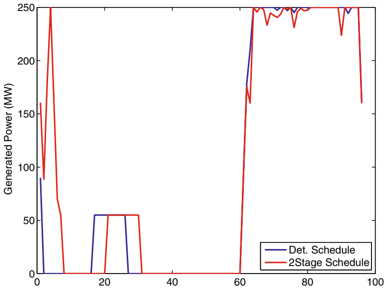
- New algo: double decomposition (by units and scenarios) using the same ingredients

Numerical illustration for stochastic unit-commitment

- On a 2013 EDF instance (medium-size)
  - deterministic problem : 50k continuous variables, 27k binary variables, 815k constraints
  - stochastic version (50 scenarios) : 1,200k continuous var., 700k binary var., 20,000k constraints
- Our method allows to solve it (in reasonable time)
- Observation: generation transferred from cheap/inflexible to expensive/flexible
- Example: production schedules for 2 units: determinist vs stochastic



cheap/inflexible unit (nuclear)



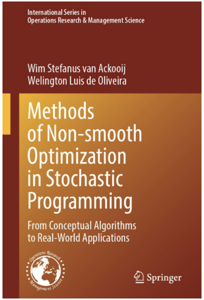
expensive/flexible unit (gaz)

Conclusion on this spotlight

- We all agree : electricity managment optimization is huge
- Nonsmoothness 1: Lagrangian decomposition
- Nonsmoothness 2: robustness against (weather) uncertainties

Conclusion on this spotlight

- We all agree : electricity managment optimization is huge
- Nonsmoothness 1: Lagrangian decomposition 
- Nonsmoothness 2: robustness against (weather) uncertainties
- You have THE expert, cf the book [van Ackooij, Oliveira '25]
- [Azema, Leclère, Van Ackooij '24]: new approach uncertain UC...
  ...by on distributionnally robust optimization



Spotlight #2: towards resilient predictions

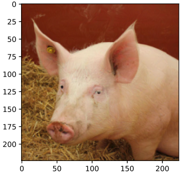
Distributionnally robust optimization

to improve fairness and resilience in machine learning

Beyond impressive results of deep learning

Don't forget how fragile deep learning can be !

Flying pigs (notebooks of NeurIPS 2018, tutorial on robustness)

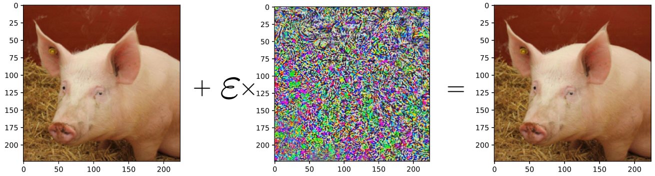


pig (99%)

Beyond impressive results of deep learning

Don't forget how fragile deep learning can be !

Flying pigs (notebooks of NeurIPS 2018, tutorial on robustness)

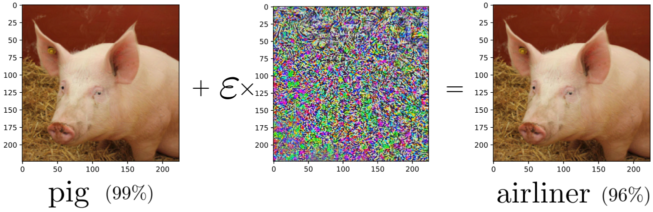


pig (99%)

Beyond impressive results of deep learning

Don't forget how fragile deep learning can be !

Flying pigs (notebooks of NeurIPS 2018, tutorial on robustness)

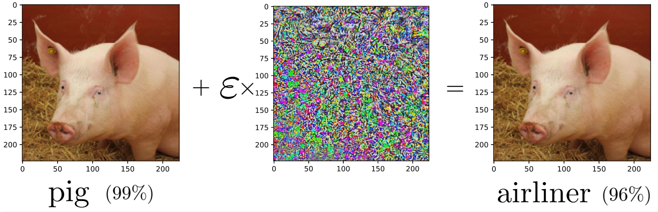


"ML is a wonderful technology: it makes pigs fly" [Kolter, Madry '18]

Beyond impressive results of deep learning

Don't forget how fragile deep learning can be !

Flying pigs (notebooks of NeurIPS 2018, tutorial on robustness)



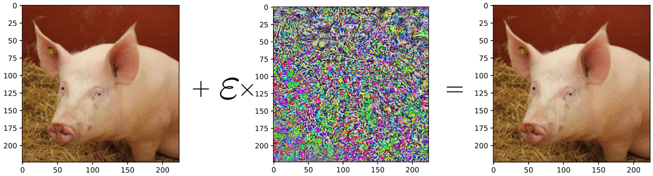
"ML is a wonderful technology: it makes pigs fly" [Kolter, Madry '18]

Attacks against self-driving cars [@ CVPR '18]

Beyond impressive results of deep learning

Don't forget how fragile deep learning can be !

Flying pigs (notebooks of NeurIPS 2018, tutorial on robustness)



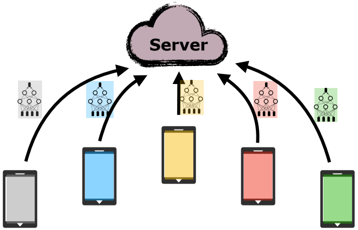
pig (99%) $+\ \mathcal{E} \times$ = airliner (96%)

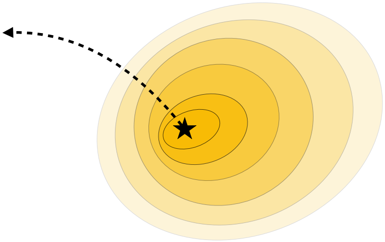
"ML is a wonderful technology: it makes pigs fly" [Kolter, Madry '18]

Attacks against self-driving cars [@ ICLR '19]

ML may perform poorly for some people

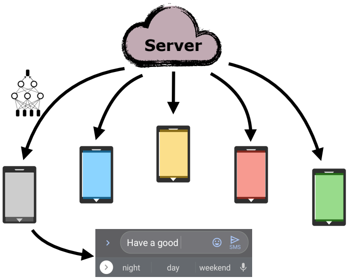
Example: Global model is trained on *average distribution* across clients (ERM)

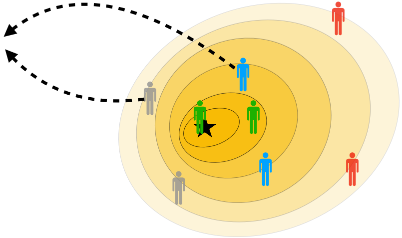




ML may perform poorly for some people

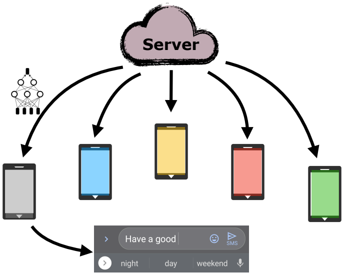
Example: Global model is deployed on *individual* clients

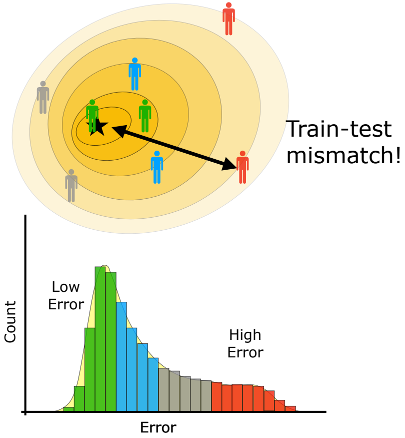




ML may perform poorly for some people

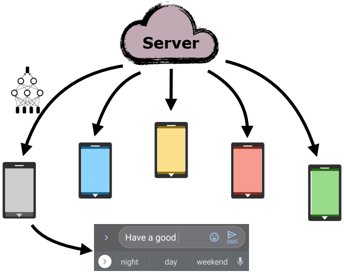
Example: Global model is deployed on *individual* clients

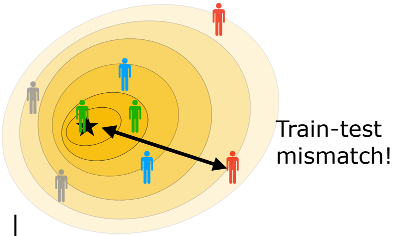


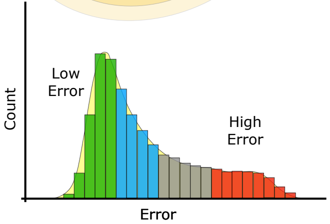


ML may perform poorly for some people

Example: Global model is deployed on *individual* clients



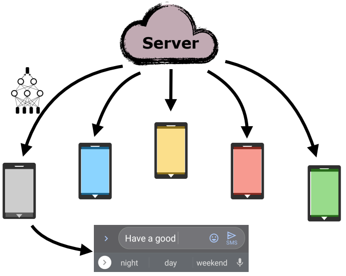


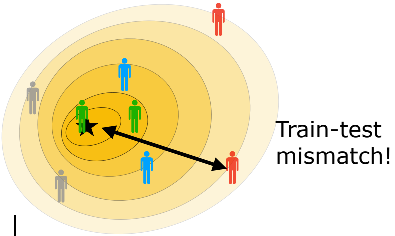


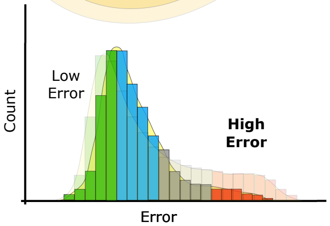
Fairness issues, e.g.

ML may perform poorly for some people

Example: Global model is deployed on *individual* clients







Fairness issues, e.g.

Amazon : l'intelligence artificielle qui n'aimait pas les femmes

Fil info

09:55 Le cri de révolte de N

09:55 Municipales à Paris: l'éponge après des "a

09:40 En Inde, des applicat intimité précieuse au

Tout

Accélérer le recrutement en faisant analyser les CV par une IA : l'idée semblait prometteuse à Amazon. Mais elle s'est mise à sous-noter les femmes candidates à des postes tech.

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica

May 23, 2016

The Washington Post

Democracy Dies in Darkness

THE ACCENT GAP

We tested Amazon's Alexa and Google's Home to see how people with accents are getting left behind in the smart-speaker revolution.

Upcoming legislation, research, and maths...

European Union has recently considered the issue

- April '19 : "Ethics Guidelines for Trustworthy AI"
- June '24 : EU Artificial Intelligence Act passed
- July '26 : High-risk AI will be required

  "Accuracy & Robustness consistently throughout their life cycle"

Upcoming legislation, research, and maths...

European Union has recently considered the issue

- April '19 : "Ethics Guidelines for Trustworthy AI"
- June '24 : EU Artificial Intelligence Act passed
- July '26 : High-risk AI will be required

  "Accuracy & Robustness consistently throughout their life cycle"

In this context, current research in my team on (distributionally) robust optimization

- is an answer to these issues and future requirements
- could be a pillar of trustworthy machine learning and decision-making
- is a nice playground for optimization, stats, and learning

Optimization set-up

- Training data: $\xi_1, \ldots, \xi_N$ (in theory: sampled from $\mathbb{P}_{\text{train}}$ unknown)

  e.g. in supervised learning: labeled data $\xi_i = (a_i, y_i)$ feature, label

- Train model: $f(x, \cdot)$ the loss function with x the parameter/decision $(\omega, \beta, \theta, \ldots)$

  e.g. least-square regression: $f\left(x, (a, y)\right) = (x^\top a - y)^2$

- Compute x via empirical risk minimization (a.k.a SAA)

  (minimize the average loss on training data)

$$\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

Optimization set-up

- Training data: $\xi_1, \ldots, \xi_N$ (in theory: sampled from $\mathbb{P}_{\text{train}}$ unknown)
  e.g. in supervised learning: labeled data $\xi_i = (a_i, y_i)$ feature, label
- Train model: $f(x, \cdot)$ the loss function with x the parameter/decision $(\omega, \beta, \theta, \ldots)$
  e.g. least-square regression: $f\left(x, (a, y)\right) = (x^\top a - y)^2$
- Compute x via empirical risk minimization (a.k.a SAA)
  (minimize the average loss on training data)

$$\min_x \; \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

- Prediction with x for different data ξ
  - Adversarial attacks, **e.g.** flying pigs, driving cakes...
  - Presence of bias, **e.g.** heterogeneous data
  - Distributional shifts: $\mathbb{P}_{\text{train}} \neq \mathbb{P}_{\text{test}}$
  - Generalization: computations with $\widehat{\mathbb{P}}_N$ and guarantees on $\mathbb{P}_{\text{train}}$

Optimization set-up

- Training data: $\xi_1, \ldots, \xi_N$ (in theory: sampled from $\mathbb{P}_{\text{train}}$ unknown)
  e.g. in supervised learning: labeled data $\xi_i = (a_i, y_i)$ feature, label

- Train model: $f(x, \cdot)$ the loss function with x the parameter/decision $(\omega, \beta, \theta, \ldots)$
  e.g. least-square regression: $f\left(x, (a, y)\right) = (x^\top a - y)^2$

- Compute x via empirical risk minimization (a.k.a SAA)
  (minimize the average loss on training data)

$$\min_x \frac{1}{N} \sum_{i=1}^{N} f(x, \xi_i)$$

- Prediction with x for different data ξ
  - Adversarial attacks, **e.g.** flying pigs, driving cakes...
  - Presence of bias, **e.g.** heterogeneous data
  - Distributional shifts: $\mathbb{P}_{\text{train}} \neq \mathbb{P}_{\text{test}}$
  - Generalization: computations with $\widehat{\mathbb{P}}_N$ and guarantees on $\mathbb{P}_{\text{train}}$

- Solution: take possible variations into account during training

...and nonsmoothness comes into play ☺

Optimization set-up

- Training data: $\xi_1, \ldots, \xi_N$ (in theory: sampled from $\mathbb{P}_{\text{train}}$ unknown)
  e.g. in supervised learning: labeled data $\xi_i = (a_i, y_i)$ feature, label

- Train model: $f(x, \cdot)$ the loss function with x the parameter/decision $(\omega, \beta, \theta, \ldots)$
  e.g. least-square regression: $f\left(x, (a, y)\right) = (x^\top a - y)^2$

- Compute x via empirical risk minimization (a.k.a SAA)
  (minimize the average loss on training data)

$$\min_x \frac{1}{N}\sum_{i=1}^N f(x, \xi_i) = \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x, \xi)] \qquad \text{with } \widehat{\mathbb{P}}_N = \frac{1}{N}\sum_{i=1}^N \delta_{\xi_i}$$

- Prediction with x for different data ξ
  - Adversarial attacks, **e.g.** flying pigs, driving cakes...
  - Presence of bias, **e.g.** heterogeneous data
  - Distributional shifts: $\mathbb{P}_{\text{train}} \neq \mathbb{P}_{\text{test}}$
  - Generalization: computations with $\widehat{\mathbb{P}}_N$ and guarantees on $\mathbb{P}_{\text{train}}$

- Solution: take possible variations into account during training

...and nonsmoothness comes into play ☺

(Wasserstein) Distributionally Robust Optimization

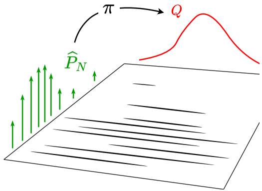
Rather than $\min_{x} \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x, \xi)]$ solve instead $\min_{x} \max_{\mathbb{Q} \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}}[f(x, \xi)]$

with $\mathcal{U}$ a neighborhood of $\widehat{\mathbb{P}}_N$

(Wasserstein) Distributionally Robust Optimization

Rather than $\min_{x} \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x,\xi)]$ solve instead $\min_{x} \max_{\mathbb{Q}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$

with $\mathcal{U}$ a neighborhood of $\widehat{\mathbb{P}}_N$



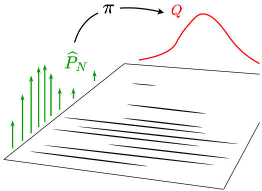
Wasserstein balls as ambiguity sets

$$\mathcal{U} = \{\ \mathbb{Q} : W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho\ \}$$

$$W(\widehat{\mathbb{P}}_N, \mathbb{Q}) = \min_{\boldsymbol{\pi}} \left\{\ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] : [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \right\}$$

(Wasserstein) Distributionally Robust Optimization

Rather than $\min_{x} \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x,\xi)]$ solve instead $\min_{x} \max_{\mathbb{Q}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$



with $\mathcal{U}$ a neighborhood of $\widehat{\mathbb{P}}_N$

Wasserstein balls as ambiguity sets

$$\mathcal{U} = \{\ \mathbb{Q} : W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho\ \}$$

$$W(\widehat{\mathbb{P}}_N, \mathbb{Q}) = \min_{\boldsymbol{\pi}} \Big\{\ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] : [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \Big\}$$

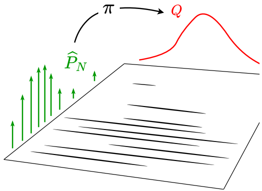
WDRO objective function for given x, $\widehat{\mathbb{P}}_N$, ρ

$$\begin{cases} \max_{\mathbb{Q}}\ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho \end{cases} \Leftrightarrow \begin{cases} \max_{\mathbb{Q},\boldsymbol{\pi}}\ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \\ \min_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] \leqslant \rho \end{cases}$$

(Wasserstein) Distributionally Robust Optimization

Rather than $\min_{x} \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x,\xi)]$ solve instead $\min_{x} \max_{\mathbb{Q}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$

with $\mathcal{U}$ a neighborhood of $\widehat{\mathbb{P}}_N$



Wasserstein balls as ambiguity sets

$$\mathcal{U} = \{\ \mathbb{Q} : W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho\ \}$$

$$W(\widehat{\mathbb{P}}_N, \mathbb{Q}) = \min_{\boldsymbol{\pi}} \left\{ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] : [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \right\}$$

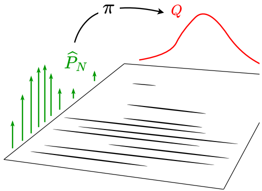
WDRO objective function for given x, $\widehat{\mathbb{P}}_N$, ρ

$$\begin{cases} \max_{\mathbb{Q}} \ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho \end{cases} \Leftrightarrow \begin{cases} \max_{\mathbb{Q},\boldsymbol{\pi}} \ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \\ \min_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] \leqslant \rho \end{cases} \Leftrightarrow \begin{cases} \max_{\boldsymbol{\pi}} \ \mathbb{E}_{[\boldsymbol{\pi}]_2}[f(x,\xi)] \\ [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N \\ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] \leqslant \rho \end{cases}$$

(Wasserstein) Distributionally Robust Optimization

Rather than $\min_{x} \ \mathbb{E}_{\widehat{\mathbb{P}}_N}[f(x,\xi)]$ solve instead $\min_{x} \ \max_{\mathbb{Q}\in\mathcal{U}} \ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$

with $\mathcal{U}$ a neighborhood of $\widehat{\mathbb{P}}_N$



Wasserstein balls as ambiguity sets

$$\mathcal{U} = \{\ \mathbb{Q} : W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho\ \}$$

$$W(\widehat{\mathbb{P}}_N, \mathbb{Q}) = \min_{\boldsymbol{\pi}} \Big\{\ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] : [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \Big\}$$

WDRO objective function for given x, $\widehat{\mathbb{P}}_N$, ρ

$$\begin{cases} \max_{\mathbb{Q}} \ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho \end{cases} \Leftrightarrow \begin{cases} \max_{\mathbb{Q},\boldsymbol{\pi}} \ \mathbb{E}_{\mathbb{Q}}[f(x,\xi)] \\ [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N, [\boldsymbol{\pi}]_2 = \mathbb{Q} \\ \min_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] \leqslant \rho \end{cases} \Leftrightarrow \begin{cases} \max_{\boldsymbol{\pi}} \ \mathbb{E}_{[\boldsymbol{\pi}]_2}[f(x,\xi)] \\ [\boldsymbol{\pi}]_1 = \widehat{\mathbb{P}}_N \\ \mathbb{E}_{\boldsymbol{\pi}}[c(\xi,\xi')] \leqslant \rho \end{cases}$$

$$\Leftrightarrow \ \min_{\lambda \geqslant 0} \ \lambda\rho + \mathbb{E}_{\widehat{\mathbb{P}}_N}\big[\max_{\xi'} \{f(x,\xi') - \lambda c(\xi,\xi')\}\big]$$

...(finite dimension) nonsmooth... computable in some (specific) cases [Kuhn *et al.* '18]

...actually many more [Vincent, Azizian, Iutzeler, Malick '24]

Illustration: gain in fairness

Federated learning framework with heterogeneous users (...) [Pillutla, Laguel, M., Harchaoui '22]

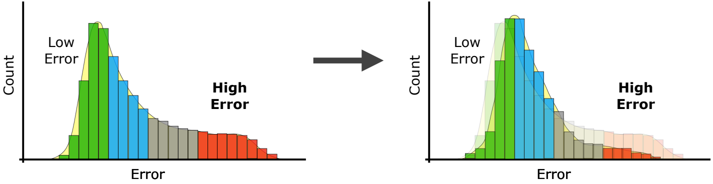
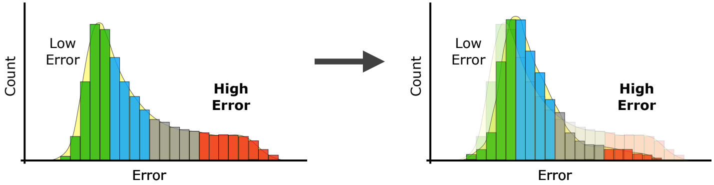


Illustration: gain in fairness

Federated learning framework with heterogeneous users (...) [Pillutla, Laguel, M., Harchaoui '22]



Experiments: (federated) classification task

ConvNet with EMNIST dataset
(1730 users, 179 images/users)

Histogram over users of test misclassif. error
Models: standard vs. robust
(dashed lines: 10%/90%-quantiles)

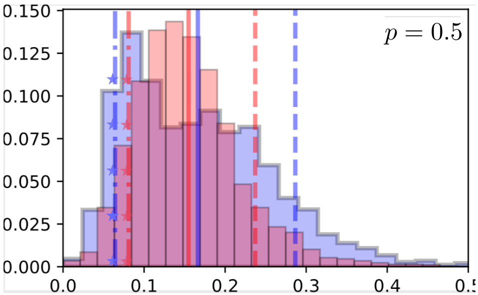
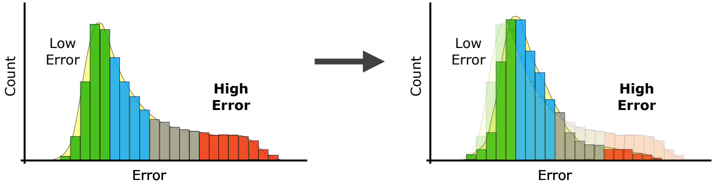


Illustration: gain in fairness

Federated learning framework with heterogeneous users (...) [Pillutla, Laguel, M., Harchaoui '22]



Experiments: (federated) classification task

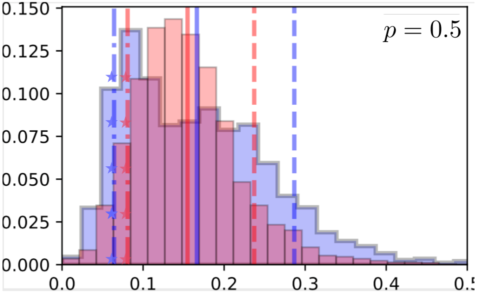
ConvNet with EMNIST dataset

(1730 users, 179 images/users)

Histogram over users of test misclassif. error

Models: standard vs. robust

(dashed lines: 10%/90%-quantiles)



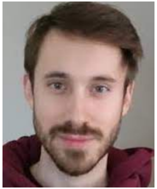
(W)DRO reshapes test histograms – towards more fairness

Main current research topic in my group

Our work

- Applications in federated learning [Laguel, Pillutla, Harchaoui, Malick '23]
- (abstract, entropic) regularizations of WDRO [Azizian, Iutzeler, Malick '22]
- Statistical guarantees [Azizian, Iutzeler, Malick '23] [Le, Malick '24]
- Numerical work for an easy-to-use toolbox `skWDRO` [Vincent, Azizian, Iutzeler, Malick '24]

Y. Laguel



F. Iutzeler

Tam Le

W. Azizian

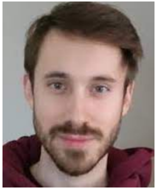
F. Vincent

Main current research topic in my group

Our work

- Applications in federated learning [Laguel, Pillutla, Harchaoui, Malick '23]
- (abstract, entropic) regularizations of WDRO [Azizian, Iutzeler, Malick '22]
- Statistical guarantees [Azizian, Iutzeler, Malick '23] [Le, Malick '24]
- Numerical work for an easy-to-use toolbox `skWDRO` [Vincent, Azizian, Iutzeler, Malick '24]

Y. Laguel



F. Iutzeler

Tam Le

W. Azizian

F. Vincent

Easy to use, with few lines of code

Scikitlearn

```python
from sklearn.linear_model import LogisticRegression # scikit-learn's standard version
from skwdro.linear_models import LogisticRegression as WDROLogisticRegression # WDRO version
```

Pytorch

```python
def main():
    device = "cuda" if pt.cuda.is_available() else "cpu"
    model = MyShallowNet([1, 50, 30, 10, 1]).to(device)

    rho = pt.tensor(1e-1).to(device)

    x = pt.sort(pt.flatten(
        pt.linspace(0., 1., 10, device=device).unsqueeze(0)\
        + pt.randn(10000, 10, device=device) * 1e-1
    ))[0]
    y = f(x) + pt.randn(100000, device=device) * 2e-2
    dataset = DataLoader(TensorDataset(x.unsqueeze(-1), y.unsqueeze(-1)), batch_size=5000, shuffle=True)

    # New line: "dualize" the loss
    dual_loss = dualize_primal_loss(
            nn.MSELoss(reduction='none'),
            model,
            rho,
            x.unsqueeze(-1),
            y.unsqueeze(-1)
        )

    model = train(dual_loss, dataset, 1000) # type: ignore
    model.eval()
```

You can easily robustify your own models with `skWDRO` !

Conclusion on this spotlight

- Deep learning works very well... unless it does not.
- Need for more robustness (resilience, fairness...) – brought by max/nonsmoothness
- Wasserstein DRO is a nice playground
- Advertizing: skWDRO

Try it out !

robustify our model with `skWDRO` !

scikitlearn interface + pytorch wrapper

A final slide

Main take-aways

- Nonsmooth optimization rocks
- Electricity managment optimization is huge

  Handling size and uncertainty leads to nonsmooth optimization
- Deep learning works very well... unless it does not

  Handling robustness leads to nonsmooth optimization
- More work is needed resilience, fairness...

A final slide

Main take-aways

- Nonsmooth optimization rocks
- Electricity managment optimization is huge

  Handling size and uncertainty leads to nonsmooth optimization
- Deep learning works very well... unless it does not

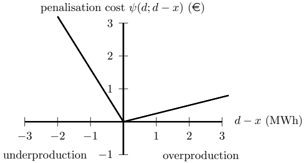
  Handling robustness leads to nonsmooth optimization
- More work is needed resilience, fairness...

thank you all ☺

Robust unit-commitment

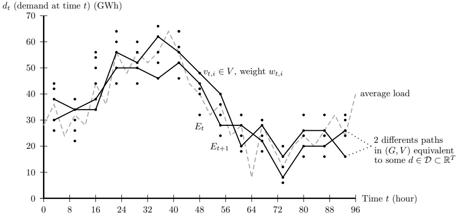
A simple robust approach
(VanAckooij Lebbe Malick '15)

- get rid of bound constraint
- penalize instead the worst gap



$$\begin{cases} \min & c^\top x + \max_{\xi \in \Xi} \sum_{t=1}^{T} \psi\big(\sum_i x_i^t - \xi^t\big) \\ x \in X & \end{cases}$$

Complex model of uncertainty set Ξ (vs Ξ finite or $\Xi = [d_{\text{min}}, d_{\text{max}}]^T$)



The model of Minoux 2012

- is finite but of high cardinality
- expresses temporal dependencies
- preserves a fast computability

Beyond flying pigs

One-pixel attack

[@ NeurIPS '19]

keep in mind how fragile deep learning techniques can be

Teapot(24.99%)
Joystick(37.39%)

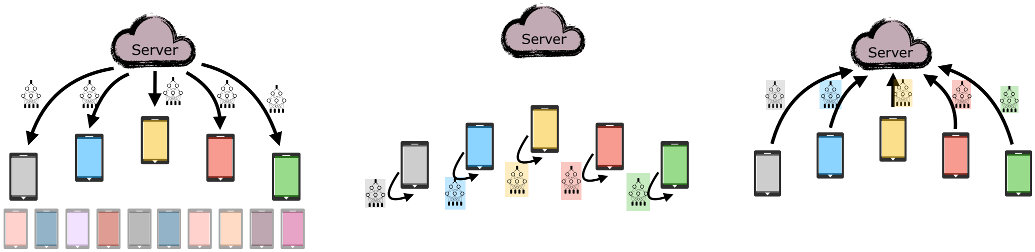
Setting: federated learning in a nutschell

- Standard learning: get all the data and learn your model on it
- Efficient... but is privacy invasive (hospitals, compagnies...)
- Idea : move the model not the data !
- Usual learning algorithm : FedAvg [McMahan *et al* 2017]
  (based on old ideas, e.g. [Mangasarian 1995])

Step 1 of 3: Server broadcasts global model to sampled clients

Step 2 of 3: Clients perform some local SGD steps on their local data

Step 3 of 3: Aggregate client updates securely

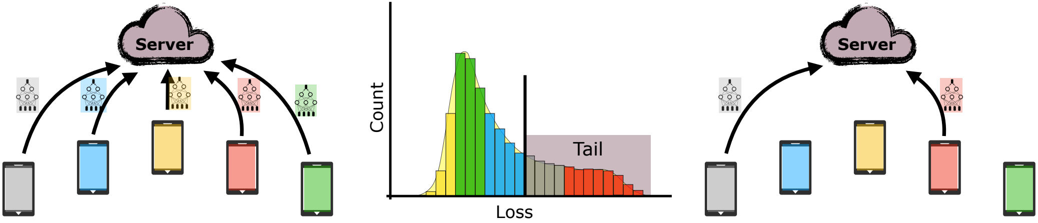


DRO/superquantile in action in federated learning

Only step 3 differs between Standard ERM approach and our DRO approach

Step 3 of 3: Aggregate updates contributed by ***all clients***

Step 3 of 3: Aggregate updates contributed by ***tail clients*** *only*



DRO approach is fully compatible with secure aggregation and differential privacy [Pillutla, Laguel, M., Harchaoui '22]

Convergence analysis

Analysis when F_i are smooth (and nonconvex)

Challenges: non-smoothness of R_θ, biais due to local participation,...

Theorem ([Pillutla, Laguel, M., Harchaoui '23])

Suppose F_i are G-Lipschitz and with gradients L-Lipshitz

$$\mathbb{E}\|\nabla \Phi_\theta^{2L}(x_t)\|^2 \leqslant \sqrt{\frac{\Delta L G^2}{t}} + (1-\tau)^{1/3}\left(\frac{\Delta L G}{t}\right)^{2/3} + \frac{\Delta L}{t}$$

with t: nb comm. rounds, τ: nb local updates, and Δ: initial error

where $\Phi_\theta^\mu(x) = \inf_y \left\{ \bar{R}_\theta(y) + \frac{\mu}{2}\|y-x\|^2 \right\}$ (Moreau♡ enveloppe) [Davis Drus. '21]

$\bar{R}_\theta$ an approximation of R_θ with unbiased gradient [Levy *et al* '21]

\+ result of linear convergence when F_i are convex (add smoothing and regularization)

WDRO objective to be minimized

Dual WDRO is nonsmooth (which complicates resolution [Kuhn *et al.* '18])

$$R_\rho(f) = \min_{\lambda \geqslant 0} \; \lambda\rho + \mathbb{E}_{\mathbb{P}}[\max_{\xi'} \{f(\xi') - \lambda\|\xi - \xi'\|^2\}]$$

What about smoothing ? Smoothed counterpart

$$R_\rho^\varepsilon(f) = \min_{\lambda \geqslant 0} \; \lambda\rho + \varepsilon\, \mathbb{E}_{\mathbb{P}} \log \left(\mathbb{E}_{\xi' \sim \pi_0(\cdot|\xi)} e^{\frac{f(\xi') - \lambda\|\xi - \xi'\|^2}{\varepsilon}} \right)$$

(Nice interpretation as entropy-regularized WDRO)

Theorem (approximation bounds for WDRO [Azizian, Iutzeler, M. '21])

Under mild assumptions (non-degeneracy, lipschitz), if the support of $\mathbb{P}$ is contained in a compact convex set $\Xi \subset \mathbb{R}^d$, then

$$0 \;\leqslant\; R_\rho(f) - R_\rho^\varepsilon(f) \;\leqslant\; \left(C\,\varepsilon\,\log\frac{1}{\varepsilon}\right) d$$

Entropic regularization: OT vs. WDRO

KL (Kullback-Leiber) divergence: $\mathrm{KL}(\mu|\nu) = \begin{cases} \int \log \frac{\mathrm{d}\mu}{\mathrm{d}\nu}\, \mathrm{d}\mu & \mu \ll \nu \\ +\infty & \text{otherwise} \end{cases}$

OT: Sinkhorn distance, very popular from [Cuturi '13]

$$\min_{\boldsymbol{\pi}} \left\{ \mathbb{E}_{\boldsymbol{\pi}}[\|\xi - \xi'\|^2] + \varepsilon\, \mathrm{KL}(\boldsymbol{\pi}|\pi_0) : \boldsymbol{\pi} \text{ with marginals } [\boldsymbol{\pi}]_1 = \mathbb{P} \text{ and } [\boldsymbol{\pi}]_2 = \mathbb{Q} \right\}$$

WDRO: entropic regularization, seemingly new [Azizian, Iutzeler, M. '21]

$$\begin{cases} \max_{\boldsymbol{\pi}} \quad \mathbb{E}_{[\boldsymbol{\pi}]_2}[f(\xi)] - \varepsilon\, \mathrm{KL}(\boldsymbol{\pi}|\pi_0) \\ \quad [\boldsymbol{\pi}]_1 = \mathbb{P} \\ \quad \mathbb{E}_{\boldsymbol{\pi}}[\|\xi - \xi'\|^2] + \delta\, \mathrm{KL}(\boldsymbol{\pi}|\pi_0) \leqslant \rho \end{cases}$$

Subtility:

in **OT**, take $\pi_0 = \mathbb{P} \otimes \mathbb{Q}$

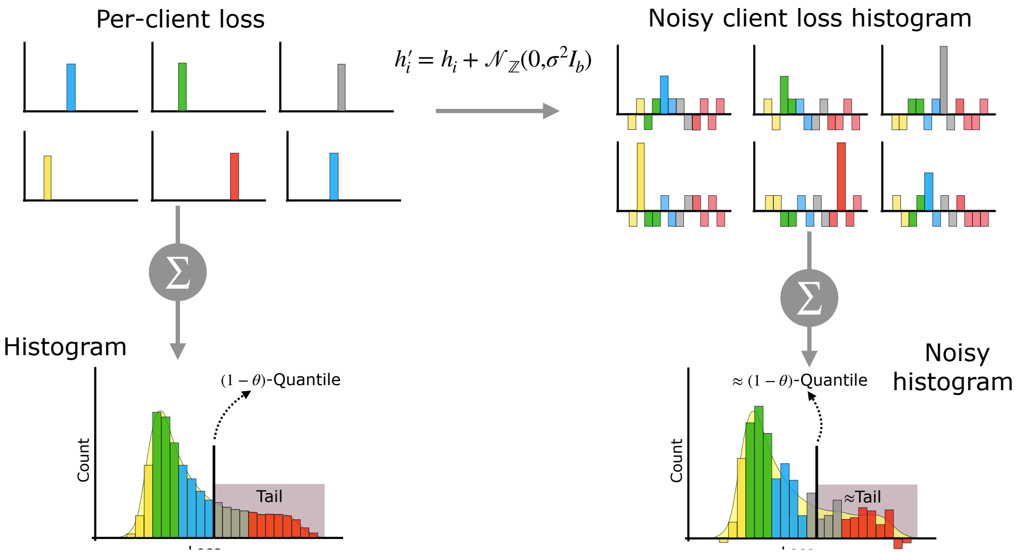
$[\boldsymbol{\pi}]_1 = \mathbb{P}, [\boldsymbol{\pi}]_2 = \mathbb{Q} \Rightarrow \boldsymbol{\pi} \ll \pi_0$

vs

but in **WDRO**, $[\pi_0]_2$ not fixed !

$\pi_0(\mathrm{d}\xi, \mathrm{d}\xi') \;\propto\; \mathbb{P}(\mathrm{d}\xi)\, \mathbb{I}_{\xi' \in \Xi}\, e^{-\frac{\|\xi - \xi'\|^2}{\sigma}}\, \mathrm{d}\xi'$

Quantile by secure aggregation



Existing statistical guarantees of WDRO

- Suppose $\xi_1, \ldots, \xi_N \sim \mathbb{P}_{\text{train}}$ (where $\xi \in \mathbb{R}^d$)
- Computations with $\widehat{\mathbb{P}}_N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\xi_i}$ and guarantees with $\mathbb{P}_{\text{train}}$?
- We manipulate the WDRO risk : $R_\rho(x) = \max_{W(\widehat{\mathbb{P}}_N, \mathbb{Q}) \leqslant \rho} \mathbb{E}_{\mathbb{Q}}[f(x, \xi)]$
- Obviously, if ρ, N large enough such that $W(\mathbb{P}_{\text{train}}, \widehat{\mathbb{P}}_N) \leqslant \rho$, then

$$\underbrace{R_\rho(x)}_{\text{can compute \& optimize}} \geqslant \underbrace{\mathbb{E}_{\mathbb{P}_{\text{train}}}[f(x, \xi)]}_{\text{cannot access}}$$

- It requires $\rho \propto 1/\sqrt[d]{N}$ [Fournier and Guillin '15] (issue)
- Not optimal: $\rho \propto 1/\sqrt{N}$ suffices
  - asymptotically [Blanchet *et al* '22]
  - in particular cases [Shafieez-Adehabadeh *et al* '19]
  - or with error terms [Gao '22]

Extended exact generalization guarantees of WDRO

Our approach: a direct "optim." approach (work to get a concentration on the dual function)

Theorem ([Azizian, Iutzeler, M. '23], [Le, M. '24])

Assumptions: parametric family $f(x,\cdot)$ + compactness on x + compactness on ξ + non-degeneracy

For $\delta \in (0,1)$, *if* $\rho \geqslant O\left(\sqrt{\frac{\log 1/\delta}{N}}\right)$ *then* *w.p.* $1-\delta$,

Generalization guarantee: $R_\rho(x) \geqslant \mathbb{E}_{\mathbb{P}_{train}}[f(x,\xi)]$

Extended exact generalization guarantees of WDRO

Our approach: a direct "optim." approach (work to get a concentration on the dual function)

Theorem ([Azizian, Iutzeler, M. '23], [Le, M. '24])

Assumptions: parametric family $f(x,\cdot)$ + compactness on x + compactness on ξ + non-degeneracy

For $\delta \in (0,1)$, *if* $\rho \geqslant O\left(\sqrt{\frac{\log 1/\delta}{N}}\right) = \rho_n$ *then* *w.p.* $1-\delta$,

Generalization guarantee: $R_\rho(x) \geqslant \mathbb{E}_{\mathbb{P}_{train}}[f(x,\xi)]$

Distribution shifts:

$$W(\mathbb{P}_{train}, \mathbb{Q})^2 \leqslant \rho(\rho - \rho_n) \quad \textit{it holds} \quad R_\rho(x) \geqslant \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$$

Extended exact generalization guarantees of WDRO

Our approach: a direct "optim." approach (work to get a concentration on the dual function)

Theorem ([Azizian, Iutzeler, M. '23], [Le, M. '24])

Assumptions: parametric family $f(x,\cdot)$ + compactness on x + compactness on ξ + non-degeneracy

For $\delta \in (0,1)$, *if* $\rho \geqslant O\left(\sqrt{\frac{\log 1/\delta}{N}}\right) = \rho_n$ *then w.p.* $1-\delta$,

Generalization guarantee: $R_\rho(x) \geqslant \mathbb{E}_{\mathbb{P}_{train}}[f(x,\xi)]$

Distribution shifts:

$$W(\mathbb{P}_{train},\mathbb{Q})^2 \leqslant \rho(\rho-\rho_n) \quad \text{it holds} \quad R_\rho(x) \geqslant \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$$

Asymptotic tightness:

$$W(\mathbb{P}_{train},\mathbb{Q})^2 \leqslant \rho(\rho+\rho_n) \quad \text{it holds} \quad R_\rho(x) \leqslant \max_{\mathbb{Q}} \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$$

Extended exact generalization guarantees of WDRO

Our approach: a direct "optim." approach (work to get a concentration on the dual function)

Theorem ([Azizian, Iutzeler, M. '23], [Le, M. '24])

Assumptions: parametric family $f(x,\cdot)$ + compactness on x + compactness on ξ + non-degeneracy

For $\delta \in (0,1)$, *if* $\rho \geqslant O\left(\sqrt{\frac{\log 1/\delta}{N}}\right) = \rho_n$ *then w.p.* $1-\delta$,

Generalization guarantee: $R_\rho(x) \geqslant \mathbb{E}_{\mathbb{P}_{train}}[f(x,\xi)]$

Distribution shifts:

$$W(\mathbb{P}_{train}, \mathbb{Q})^2 \leqslant \rho(\rho - \rho_n) \quad \textit{it holds} \quad R_\rho(x) \geqslant \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$$

Asymptotic tightness:

$$W(\mathbb{P}_{train}, \mathbb{Q})^2 \leqslant \rho(\rho + \rho_n) \quad \textit{it holds} \quad R_\rho(x) \leqslant \max_{\mathbb{Q}} \mathbb{E}_{\mathbb{Q}}[f(x,\xi)]$$

- Universal result: deep learning, kernels, family of invertible mappings (e.g. normalizing flows)
- Retrieve existing results in linear/logistic regressions [Shafieez-Adehabadeh *et al* '19]

Numerical optimization

Smoothed dual WDRO problem: minimizing a differentiable objective function

$$\min_{x} \min_{\lambda \geqslant 0} \lambda\rho + \frac{1}{N}\sum_{i=1}^{N} \varepsilon \log\left(\mathbb{E}_{\xi' \sim \mathcal{N}(\xi_i, \sigma^2)} \exp\left(\frac{f(x,\xi') - \lambda\|\xi - \xi'\|^2}{\varepsilon}\right)\right)$$

Our approach: use Pytorch tools (automatic backward diff. & adaptive SDG-like methods)

Numerical optimization

Smoothed dual WDRO problem: minimizing a differentiable objective function

$$\min_{x} \min_{\lambda \geqslant 0} \lambda\rho + \frac{1}{N}\sum_{i=1}^{N} \varepsilon \log\left(\mathbb{E}_{\xi' \sim \mathcal{N}(\xi_i, \sigma^2)} \exp\left(\frac{f(x,\xi') - \lambda\|\xi - \xi'\|^2}{\varepsilon}\right)\right)$$

Our approach: use Pytorch tools (automatic backward diff. & adaptive SDG-like methods)

Not so easy, because of the inner expectation...

Numerical optimization

Smoothed dual WDRO problem: minimizing a differentiable objective function

$$\min_{x} \min_{\lambda \geqslant 0} \quad \lambda\rho + \frac{1}{N}\sum_{i=1}^{N} \varepsilon \log\left(\mathbb{E}_{\xi' \sim \mathcal{N}(\xi_i,\sigma^2)} \exp\left(\frac{f(x,\xi') - \lambda\|\xi - \xi'\|^2}{\varepsilon}\right)\right)$$

Our approach: use Pytorch tools (automatic backward diff. & adaptive SDG-like methods)

Not so easy, because of the inner expectation...

Requires some (hard) work on computational aspects, e.g.

- Control the biais of the lower bound, after sampling $\xi'_j \sim \mathcal{N}(\xi_i, \sigma^2)$

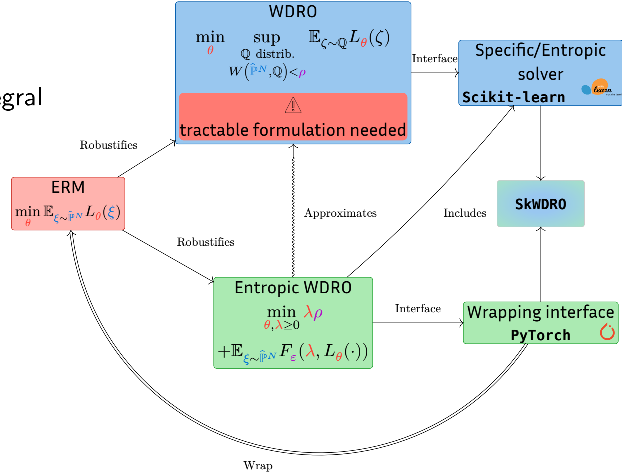
$$\min_{x} \min_{\lambda \geqslant 0} \quad \lambda\rho + \frac{1}{N}\sum_{i=1}^{N} \varepsilon \log\left(\frac{1}{M}\sum_{j=1}^{M} \exp\left(\frac{f(x,\xi'_j) - \lambda\|\xi - \xi'_j\|^2}{\varepsilon}\right)\right)$$

  Objective still sharply peaked (so high variance in the gradient estimate...)

- Use importance sampling: sample the ξ'_j shifted towards the gradient

Python Toolbox `skWDRO`

- Control on the approximations
- Importance sampling for the inner integral
- Careful logsumexp
- Numerically stable backward pass
- Heuristics to set ε and σ
- Efficient heuristic to set starting λ
- All-in-one API
- User-friendly interfaces (Pytorch and Scikitlearn)



Try it out !

More in [Vincent, Azizian, Iutzeler, M. '24]