Structured linear models: contributions from statistics and optimization

Joseph Salmon

http://josephsalmon.eu

IMAG
Univ. Montpellier
CNRS

Contact:

Joseph Salmon

joseph.salmon@umontpellier.fr

http://josephsalmon.eu

Github: @josephsalmon

Twitter: @salmonjsph

Joint works with various colleagues

Quentin Bertrand (INRIA, Parietal Team)

Quentin Klopfenstein (Univ. Bourgogne Franche-Comté)

Samuel Vaiter (Univ. Bourgogne Franche-Comté, CNRS)

Mathieu Blondel (Google Research, Brain team)

Mathurin Massias (University of Genova)

Eugene Ndiaye (Riken, Tokyo)

Alexandre Gramfort (INRIA, Parietal Team)

Olivier Fercoq (Télécom Paris)

Sophie Lèbre (Univ. Paul Valéry Montpellier 3, IMAG)

Florent Bascou (Univ. Montpellier, IMAG)

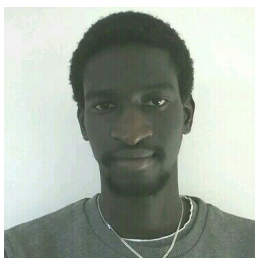
Quentin B.

Quentin K.

Mathieu

Samuel

Mathurin



Eugene

Alexandre

Olivier

Sophie

Florent

Overview

Motivation / Examples

Variable selection and sparsity

Algorithms for solving non-smooth convex problems

Hyperoptimization

Extensions to general structures

Outline

Motivation / Examples

Variable selection and sparsity

Algorithms for solving non-smooth convex problems

Hyperoptimization

Extensions to general structures

Classification in bio-statistics

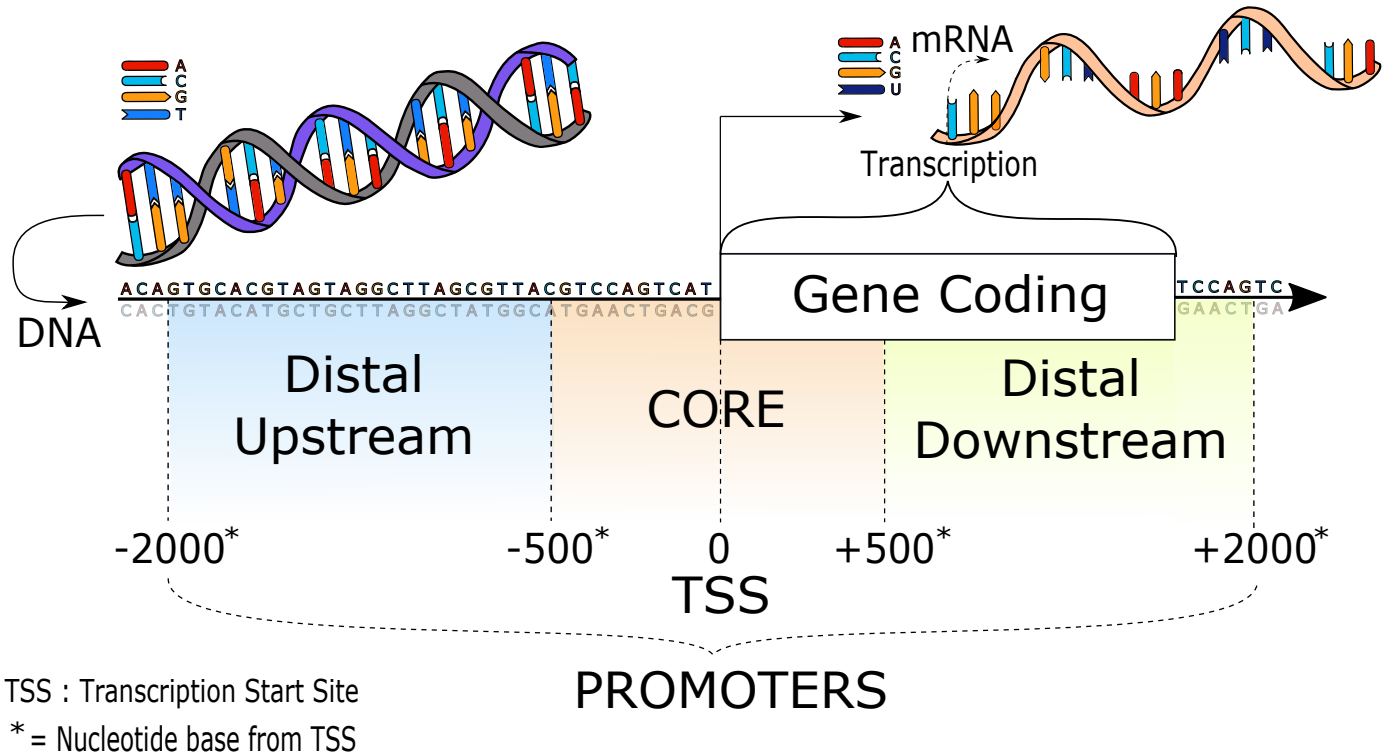
"47 patients with acute lymphoblastic leukemia (ALL) and 25 patients with acute myeloid leukemia (AML). Each of the 72 patients had bone marrow samples obtained at the time of diagnosis. The observations have been assayed with Affymetrix Hgu6800 chips, resulting in 7129 gene expressions (Affymetrix probes)."[1]

- $n = 72$ (samples)
- $p =$ **7129** (features / covariates / exploratory variables)
- $q = 2$ classes (binary classification: $+1 =$ sick / -1 = not sick)

https://github.com/ramhiser/datamicroarray/wiki/Golub-(1999)

[1] T. R. Golub et al. "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.". In: *Science* 286.5439 (1999), pp. 531–537.

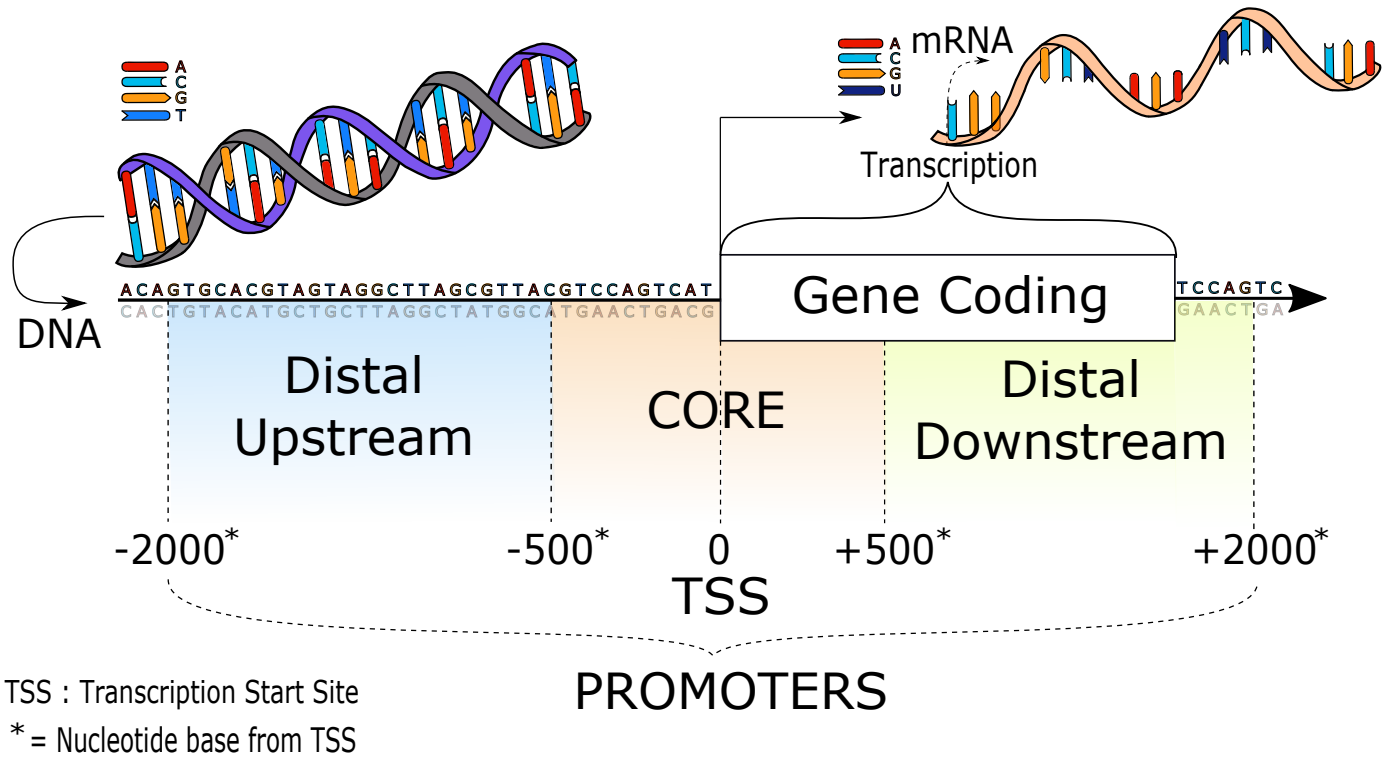
Regression in bio-statistics



- explain mRNA count $(\mathbf{y})$ from DNA sequence (X)[2]
- X : frequency of 1-letter and 2-letter words (AA, AC, AG, ...)

[2] C. Bessière et al. "Probing instructions for expression regulation in gene nucleotide compositions". In: *PLOS Computational Biology* 14.1 (Jan. 2018), pp. 1–28.

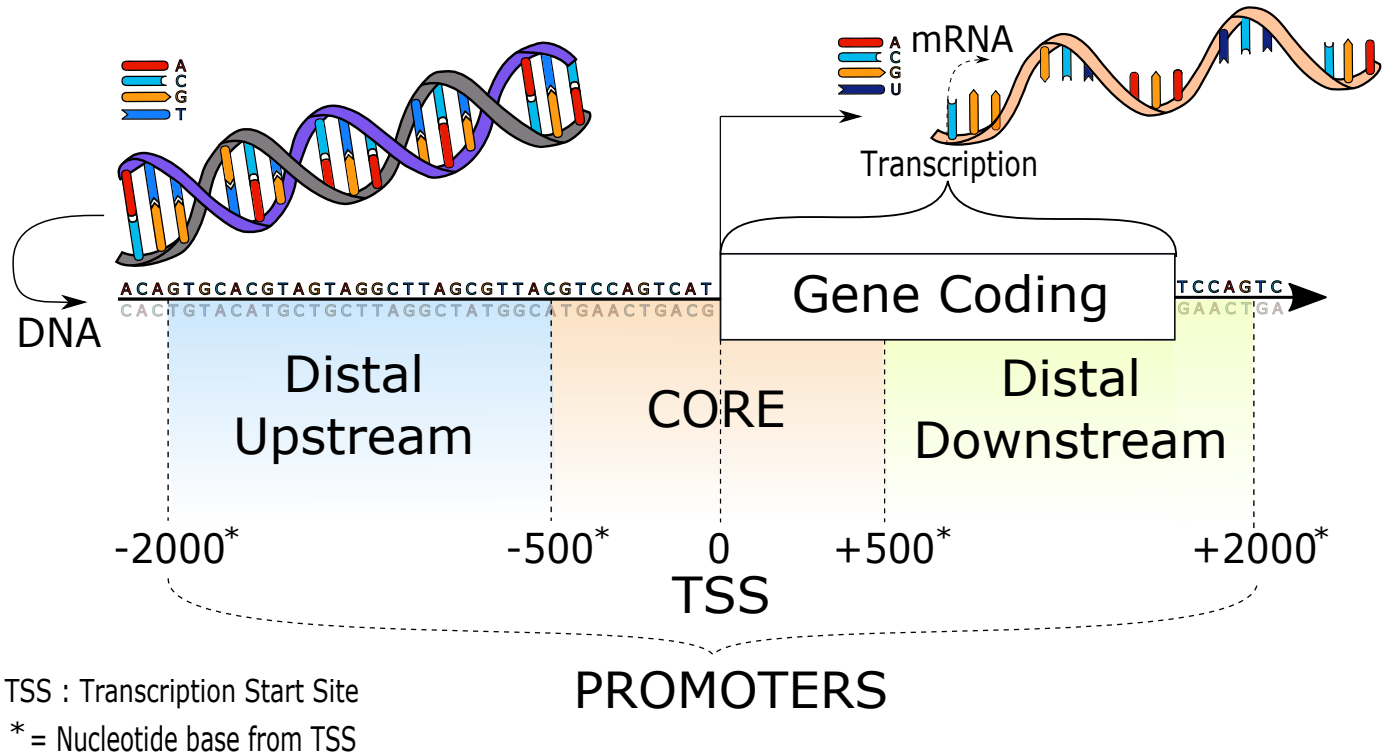
Regression in bio-statistics



- explain mRNA count ($\mathbf{y}$) from DNA sequence (X)[2]
- X : frequency of 1-letter and 2-letter words (AA, AC, AG, ...)
- With (2nd order) interactions: $n \approx \mathbf{20\ 000}$: genes number, $p \approx 500$: main features or $p \approx \mathbf{140\ 000}$ quadratic features

[2] C. Bessière et al. "Probing instructions for expression regulation in gene nucleotide compositions". In: *PLOS Computational Biology* 14.1 (Jan. 2018), pp. 1–28.

Regression in bio-statistics

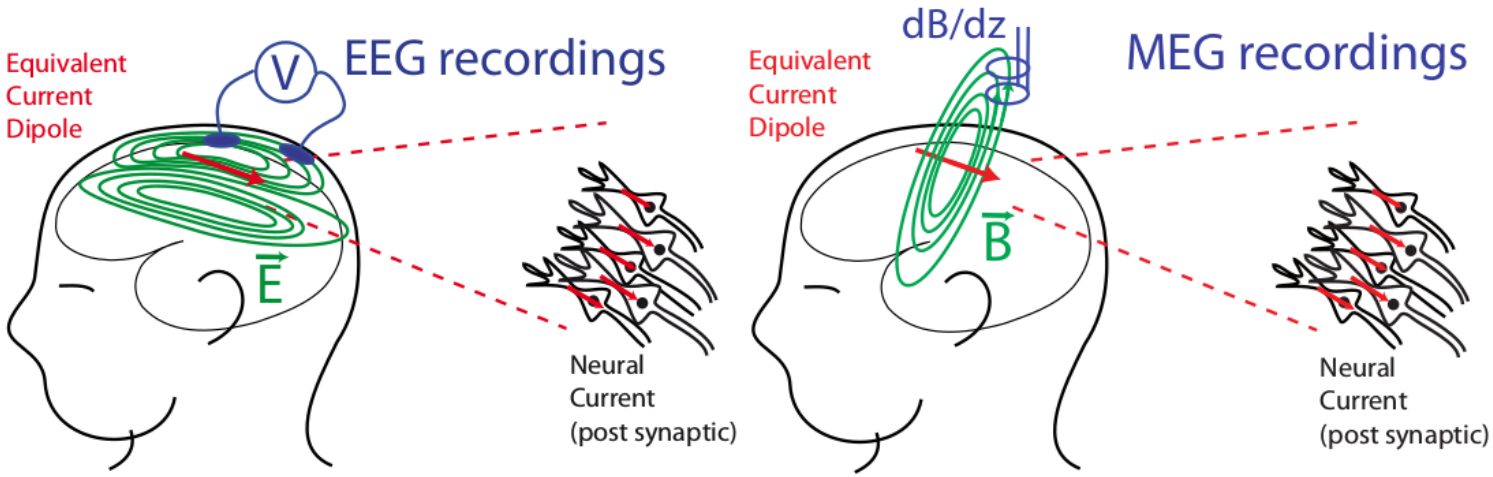


- explain mRNA count $(\mathbf{y})$ from DNA sequence (X)[2]
- X : frequency of 1-letter and 2-letter words (AA, AC, AG, ...)
- With (2nd order) interactions: $n \approx \mathbf{20\ 000}$: genes number,
  $p \approx 500$: main features or $p \approx \mathbf{140\ 000}$ quadratic features

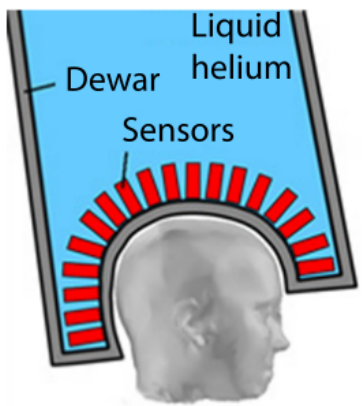
[2] C. Bessière et al. "Probing instructions for expression regulation in gene nucleotide compositions". In: *PLOS Computational Biology* 14.1 (Jan. 2018), pp. 1–28.

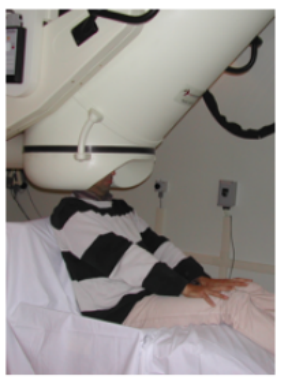
Inverse problem for neuro-imaging: M/EEG

- sensors: magneto- and electro-encephalograms (M/EEG) measures during a cognitive experiment
- sources: positions in the brain



First EEG recordings in 1929 by H. Berger



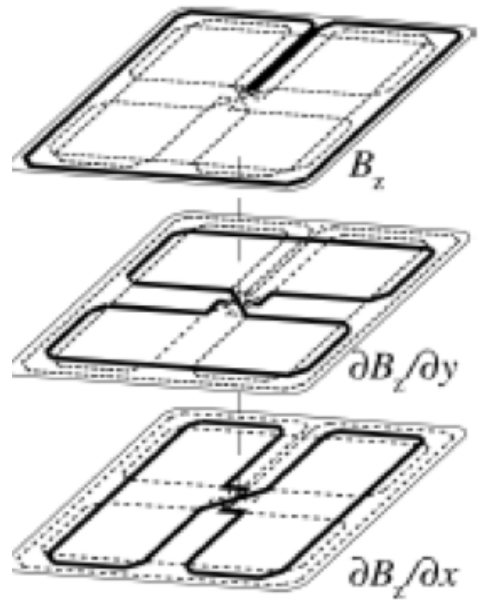


Hôpital La Timone
Marseille, France

MEG Sensors: magnet-meters et gradio-meters

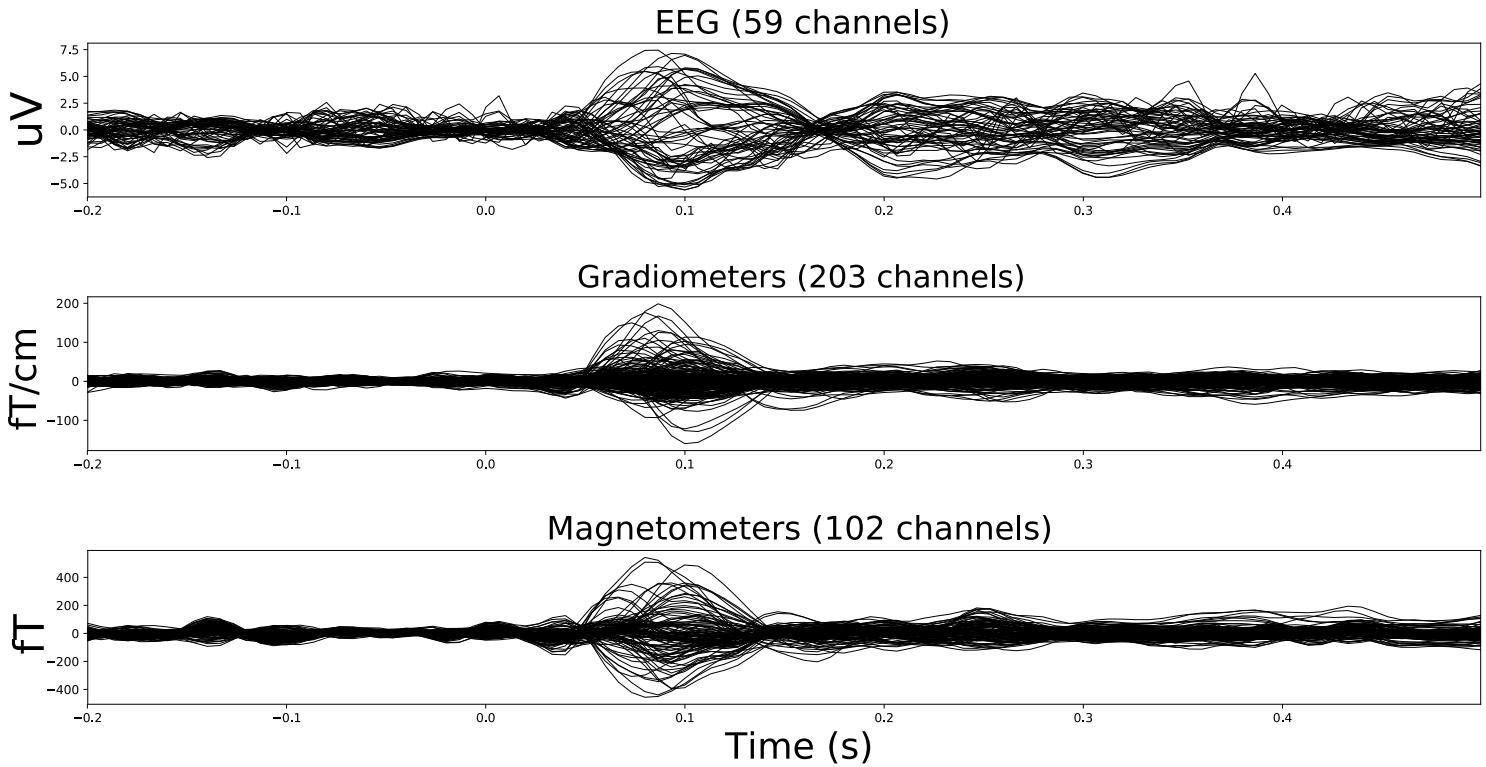
Device

Sensors

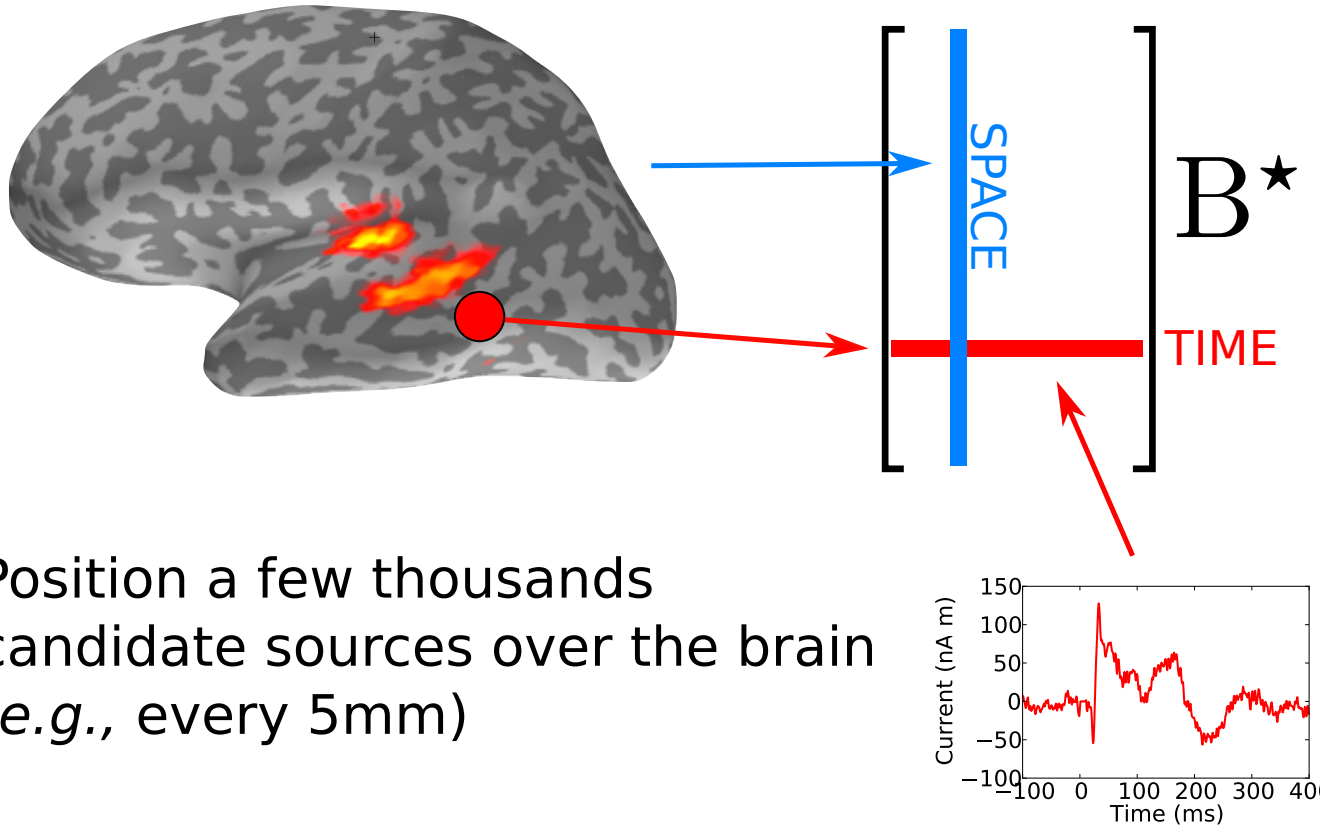


Sensors details

M/EEG data: different types of signals



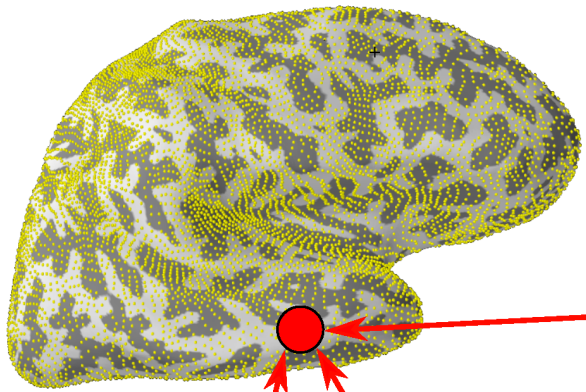
Sources model



Position a few thousands candidate sources over the brain (*e.g.*, every 5mm)

$$\mathrm{B}^\star \in \mathbb{R}^{p \times q}$$

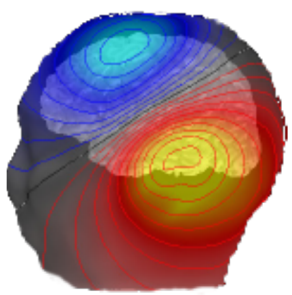
Design matrix - forward operator

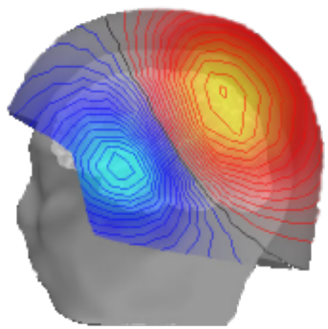


$$X = \left[\begin{array}{c|c} X_{\text{EEG}} & \\ \hline & X_{\text{MEG}} \end{array}\right] \in \mathbb{R}^{n \times p}$$

EEG:
Forward field of the electrodes

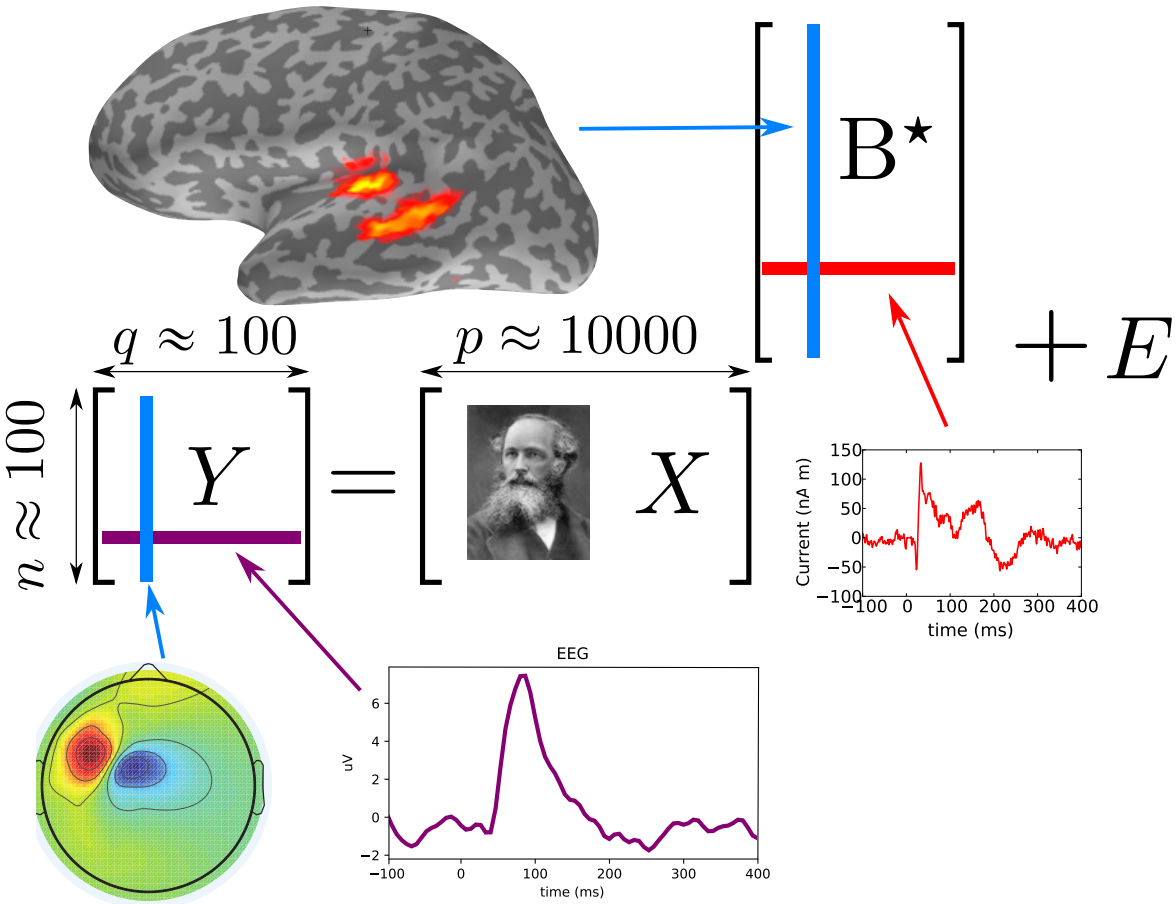
MEG:
Forward field of sensor





X: gain matrix / forward operator obtained by Maxwell's equations

Mutli-task regression



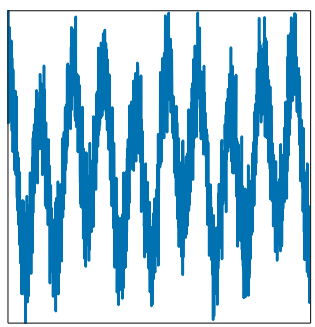
Standard dimensions:

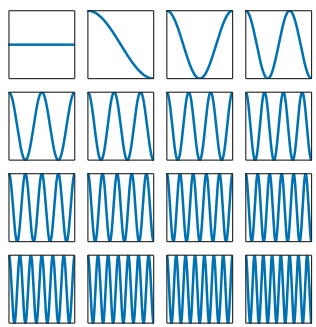
- $n = 302$ sensors
- $p = 7498$ sources (discretization in space)
- $q = 181$ time instants

Sparsity is all around

Signals : often represented combining a few **atoms** / **features**

- Fourier decomposition for sounds

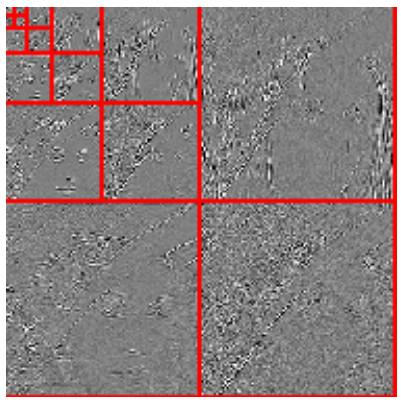




Sparsity is all around

Signals : often represented combining a few **atoms** / **features**

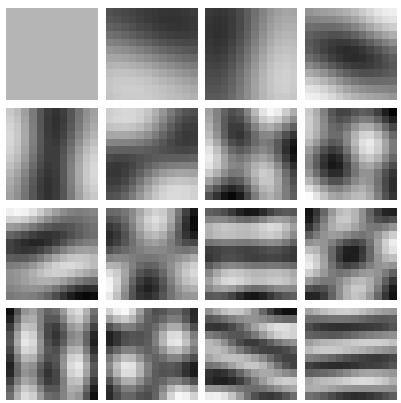
- Fourier decomposition for sounds
- Wavelet for images (1990's)



Sparsity is all around

Signals : often represented combining a few **atoms** / **features**

- Fourier decomposition for sounds
- Wavelet for images (1990's)
- Dictionary learning for images (late 2000's)



Simplest model: standard sparse regression

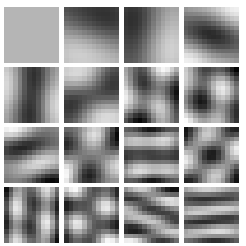
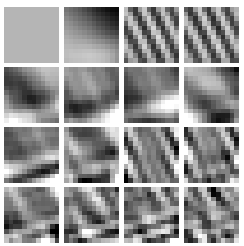
$\mathbf{y} \in \mathbb{R}^n$: signal

n samples

Simplest model: standard sparse regression

$\mathbf{y} \in \mathbb{R}^n$: signal
n samples

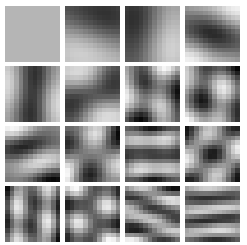
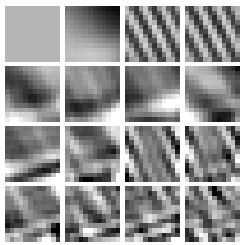

$X = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
dictionary of atoms / features / covariates
p features

Simplest model: standard sparse regression

$\mathbf{y} \in \mathbb{R}^n$: signal
n samples


$X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
dictionary of atoms / features / covariates
p features

Assumption :
signal well approximated by a **sparse** combination of atoms $\boldsymbol{\beta}^\star \in \mathbb{R}^p$
(many $j's$ with $\beta_j^\star = 0$)

$$\underbrace{\begin{bmatrix} \\ \mathbf{y} \\ \\ \end{bmatrix}}_{\mathbf{y} \in \mathbb{R}^n} \approx \underbrace{\left[\begin{array}{c|c|c} & & \\ \mathbf{x}_1 & \dots & \mathbf{x}_p \\ & & \end{array}\right]}_{X \in \mathbb{R}^{n \times p}} \cdot \underbrace{\begin{bmatrix} \beta_1^\star \\ \vdots \\ \beta_p \star \end{bmatrix}}_{\boldsymbol{\beta} \in \mathbb{R}^p}$$

$$\mathbf{y} \approx \sum_{j=1}^{p} \beta_j^\star \mathbf{x}_j \approx X \boldsymbol{\beta}^\star$$

Associated Gaussian noise model

$$\boxed{\mathbf{y} = X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon} \in \mathbb{R}^n}$$

- $\mathbf{y} \in \mathbb{R}^n$: signal / observations
- $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n \sim \mathcal{N}(0, \sigma^2)$: Gaussian **noise** with variance σ^2
- $\boldsymbol{\beta}^\star \in \mathbb{R}^p$: true parameter to recover

Context: p large w.r.t. n, sparse $\boldsymbol{\beta}^\star$

Associated Gaussian noise model

$$\boxed{\mathbf{y} = X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon} \in \mathbb{R}^n}$$

- $\mathbf{y} \in \mathbb{R}^n$: signal / observations
- $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n \sim \mathcal{N}(0, \sigma^2)$: Gaussian **noise** with variance σ^2
- $\boldsymbol{\beta}^\star \in \mathbb{R}^p$: true parameter to recover

Context: p large w.r.t. n, sparse $\boldsymbol{\beta}^\star$

Multiple objective(s):

get an estimator $\hat{\beta}$ s.t.

- **Estimation**: $\hat{\beta} \approx \beta^\star$
- **Prediction**: $X\hat{\beta} \approx X\beta^\star$
- **Support recovery**: $\mathrm{supp}(\hat{\beta}) \approx \mathrm{supp}(\beta^\star)$ (non-zero coordinates)

Associated Gaussian noise model

$$\boxed{\mathbf{y} = X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon} \in \mathbb{R}^n}$$

- $\mathbf{y} \in \mathbb{R}^n$: signal / observations
- $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n \sim \mathcal{N}(0, \sigma^2)$: Gaussian **noise** with variance σ^2
- $\boldsymbol{\beta}^\star \in \mathbb{R}^p$: true parameter to recover

Context: p large w.r.t. n, sparse $\boldsymbol{\beta}^\star$

Multiple objective(s):
get an estimator $\hat{\boldsymbol{\beta}}$ s.t.

- **Estimation**: $\hat{\boldsymbol{\beta}} \approx \boldsymbol{\beta}^\star$
- **Prediction**: $X\hat{\boldsymbol{\beta}} \approx X\boldsymbol{\beta}^\star$
- **Support recovery**: $\mathrm{supp}(\hat{\boldsymbol{\beta}}) \approx \mathrm{supp}(\boldsymbol{\beta}^\star)$ (non-zero coordinates)

Rem: extensions for classification, multi-task, etc., (end of talk)

Associated Gaussian noise model

$$\boxed{\mathbf{y} = X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon} \in \mathbb{R}^n}$$

- $\mathbf{y} \in \mathbb{R}^n$: signal / observations
- $X = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$: design matrix
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n \sim \mathcal{N}(0, \sigma^2)$: Gaussian **noise** with variance σ^2
- $\boldsymbol{\beta}^\star \in \mathbb{R}^p$: true parameter to recover

Context: p large w.r.t. n, sparse $\boldsymbol{\beta}^\star$

Multiple objective(s):

get an estimator $\hat{\boldsymbol{\beta}}$ s.t.

- **Estimation**: $\hat{\boldsymbol{\beta}} \approx \boldsymbol{\beta}^\star$
- **Prediction**: $X\hat{\boldsymbol{\beta}} \approx X\boldsymbol{\beta}^\star$
- **Support recovery**: $\mathrm{supp}(\hat{\boldsymbol{\beta}}) \approx \mathrm{supp}(\boldsymbol{\beta}^\star)$ (non-zero coordinates)

Rem: extensions for classification, multi-task, etc., (end of talk)

Outline

Motivation for sparse models

Estimators $\hat{\beta}$ of $\beta^\star$ with many zero coefficients are useful for:

- **Interpretation**: interest for practitioners
- **Theory**: counter curse of dimensionality limits
- **Computational efficiency**: especially for huge p

Underlying idea: **variable selection**

Support and ℓ_0 pseudo-norm

Definitions

ℓ_0 **pseudo-norm** of $\boldsymbol{\beta} \in \mathbb{R}^p$: number of non-zero coordinates

$$\|\boldsymbol{\beta}\|_0 = \mathrm{card}\{j \in [\![1, p]\!], \beta_j \neq 0\}$$

Rem: $\|\cdot\|_0$ is **not a norm**, $\forall t \in \mathbb{R}^*, \|t\boldsymbol{\beta}\|_0 = \|\boldsymbol{\beta}\|_0$

Rem: $\|\cdot\|_0$ it is **not even convex**, *e.g.*,
$\boldsymbol{\beta}_1 = (1, 0, 1)$, $\boldsymbol{\beta}_2 = (0, 1, 1)$ and $3 = \|\frac{\boldsymbol{\beta}_1 + \boldsymbol{\beta}_2}{2}\|_0 \geq \frac{\|\boldsymbol{\beta}_1\|_0 + \|\boldsymbol{\beta}_2\|_0}{2} = 2$

Outline

The ℓ_0 penalty

First attempt: promote sparsity using ℓ_0 as a penalty/regularization

$$\hat{\beta}_\lambda^{\ell_0} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left(\underbrace{\frac{1}{2n}\|\mathbf{y} - X\beta\|_2^2}_{\text{data fitting}} + \underbrace{\lambda \|\beta\|_0}_{\text{regularization}} \right)$$

Combinatorial problem!!!

Exact/naive resolution : consider all sub-models, *i.e.*, compute 2^p least squares computation (*i.e.*, 2^p possible supports); **NP-hard**[3]

Example:
$p = 10$: $\approx 10^3$ least squares
$p = 30$: $\approx 10^{10}$ least squares

Rem: mixed integer programming fine for small problems[4]

---

[3] B. K. Natarajan. "Sparse approximate solutions to linear systems". In: *SIAM J. Comput.* 24.2 (1995), pp. 227–234.

[4] D. Bertsimas, A. King, and R. Mazumder. "Best subset selection via a modern optimization lens". In: *Ann. Statist.* 44.2 (2016), pp. 813–852.

Statistically optimal for sparse signals (minimax sense)

Theorem[5]

For $\hat{\beta}_\lambda^{\ell_0}$ with a well chosen parameter λ (and a constant C):

$$\mathbb{E}\left(\frac{\|X\hat{\beta}_\lambda^{\ell_0} - X\beta^\star\|^2}{n}\right) \leq C\frac{\sigma^2\,\|\beta^\star\|_0}{n}\log\left(\frac{ep}{\|\beta^\star\|_0}\right)$$

Rem: least-squares prediction error $O\left(\frac{\sigma^2 p}{n}\right)$

Rem: cannot be improved (minimax sense), optimal rate[6]

[5] F. Bunea, A. B. Tsybakov, and M. H. Wegkamp. "Aggregation for Gaussian regression". In: *Ann. Statist.* 35.4 (2007), pp. 1674–1697.

[6] A. B. Tsybakov. "Optimal Rates of Aggregation". In: *COLT*. 2003, pp. 303–313.

Computationally "tractable" alternatives

- **Correlation screening**: remove the $\mathbf{x}_j$'s whose correlation with observation $\mathbf{y}$ is weak, fast (+++), intuitive (+++) but weak theory (- - -), neglect variables interactions (- - -)
- **Greedy methods**: forward/stage-wise[7],[8],[9], fast(++), intuitive(++), propagates wrong selection (- -), weak theory(-)
- **Penalized methods**
  - **convex**
  - non-convex
- **Approximate Message Passing**[10](AMP), graphical models, hard to solve (- -), theory (claimed better?)

---

[7] M. A. Efroymson. "Multiple regression analysis". In: *Mathematical methods for digital computers*. New York: Wiley, 1960, pp. 191–203.

[8] S. Mallat and Z. Zhang. "Matching Pursuit With Time-Frequency Dictionaries". In: *IEEE Trans. Image Process.* 41 (1993), pp. 3397–3415.

[9] T. Zhang. "Adaptive forward-backward greedy algorithm for learning sparse representations". In: *IEEE Trans. Inf. Theory* 57.7 (2011), pp. 4689–4708.

[10] D. L. Donoho, A., and A. Montanari. "Message-passing algorithms for compressed sensing". In: *Proceedings of the National Academy of Sciences* 106.45 (2009), pp. 18914–18919.

Outline

Lasso: penalty point of view[11]

Lasso: *Least Absolute Shrinkage and Selection Operator*

$$\hat{\boldsymbol{\beta}}^{(\lambda)} \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left(\underbrace{\frac{1}{2n}\|\mathbf{y} - X\boldsymbol{\beta}\|_2^2}_{\text{data fitting}} + \underbrace{\lambda\|\boldsymbol{\beta}\|_1}_{\text{regularization}} \right)$$

where $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$, $\lambda > 0$ controls the sparsity level

- Limiting cases: $\lim_{\lambda\to 0} \hat{\boldsymbol{\beta}}^{(\lambda)} = \hat{\boldsymbol{\beta}}^{\mathrm{LS}}$ (with smallest ℓ_1 norm)

$$\lim_{\lambda\to+\infty} \hat{\boldsymbol{\beta}}^{(\lambda)} = 0 \in \mathbb{R}^p$$

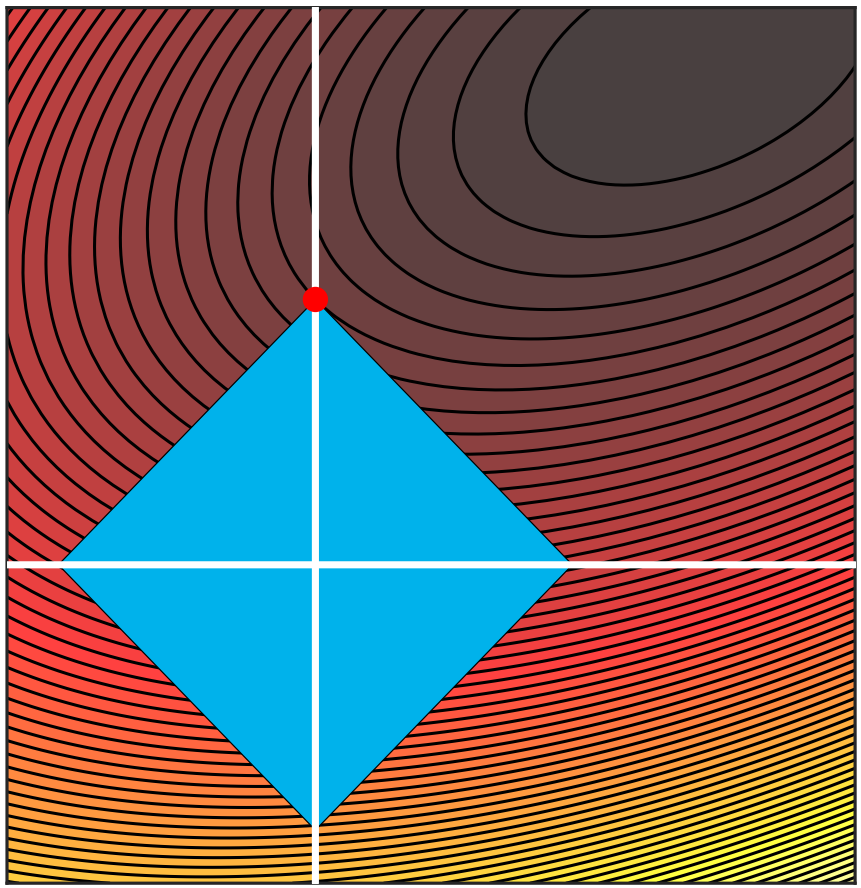
Beware: uniqueness non mandatory (*e.g.*, case $\mathbf{x}_1 = \mathbf{x}_2$)

---

[11] R. Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *J. R. Stat. Soc. Ser. B Stat. Methodol.* 58.1 (1996), pp. 267–288.

Zeroing coefficients: a vizualisation

$$\underset{\beta\in\mathbb{R}^p}{\arg\min}\ \frac{1}{2n}\|\mathbf{y}-X\beta\|_2^2+\lambda\|\beta\|_1 \iff \begin{cases}\underset{\beta\in\mathbb{R}^p}{\arg\min}\ \|\mathbf{y}-X\beta\|_2^2 \\ \text{s.t. } \|\beta\|_1\leq\gamma\end{cases}, \text{ for a } \gamma>0$$



ℓ_1 constraint : ~~non~~ sparse solution

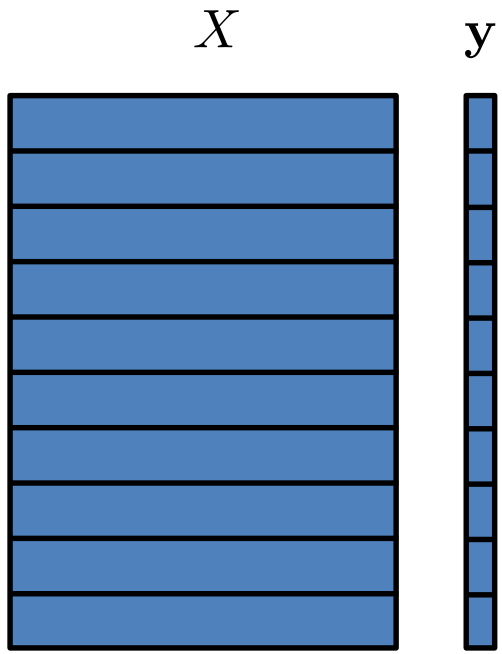
Computational bottleneck

Reminder: $\hat{\boldsymbol{\beta}}^{(\lambda)} \in \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{2n} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$

1. compute a Lasso solution (**non-smooth convex** problem)
2. tune λ using cross-validation (solving many Lasso problems)

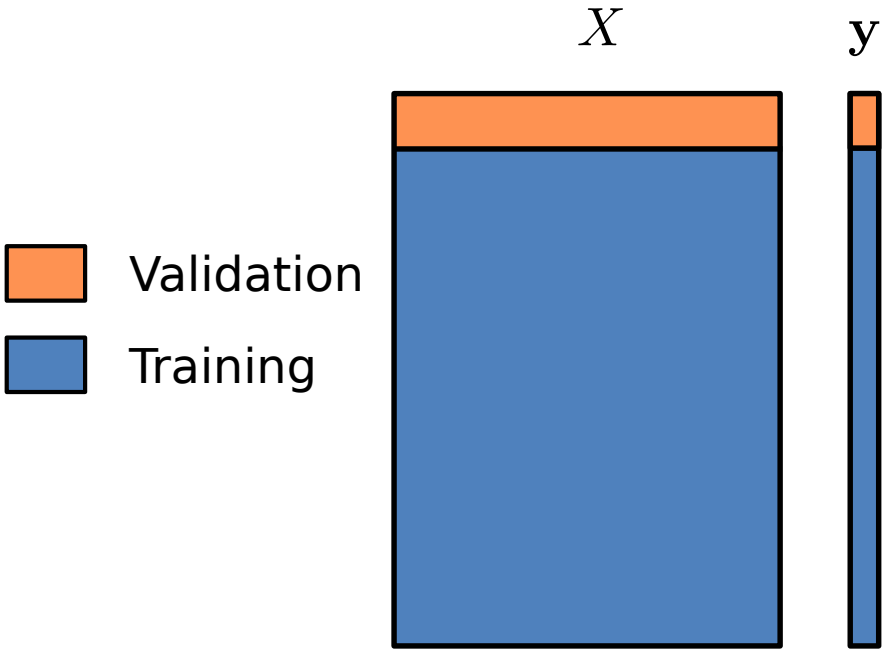
K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

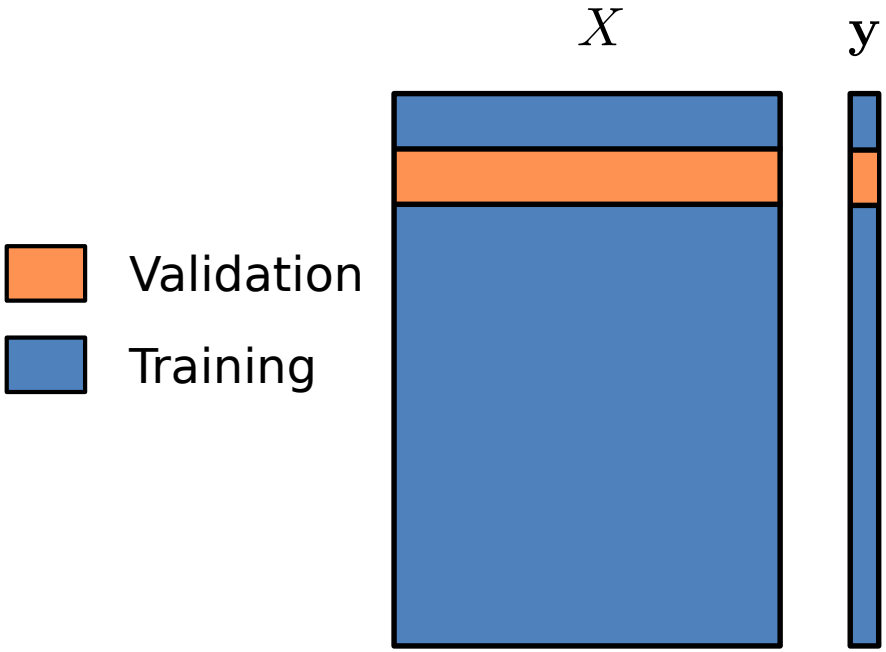


$k = 1$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

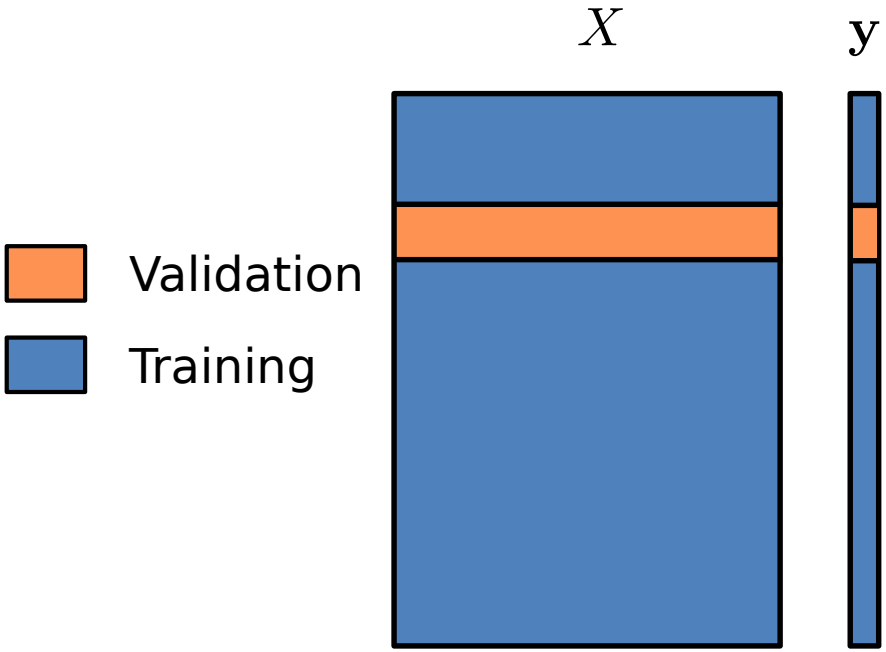


$k = 2$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

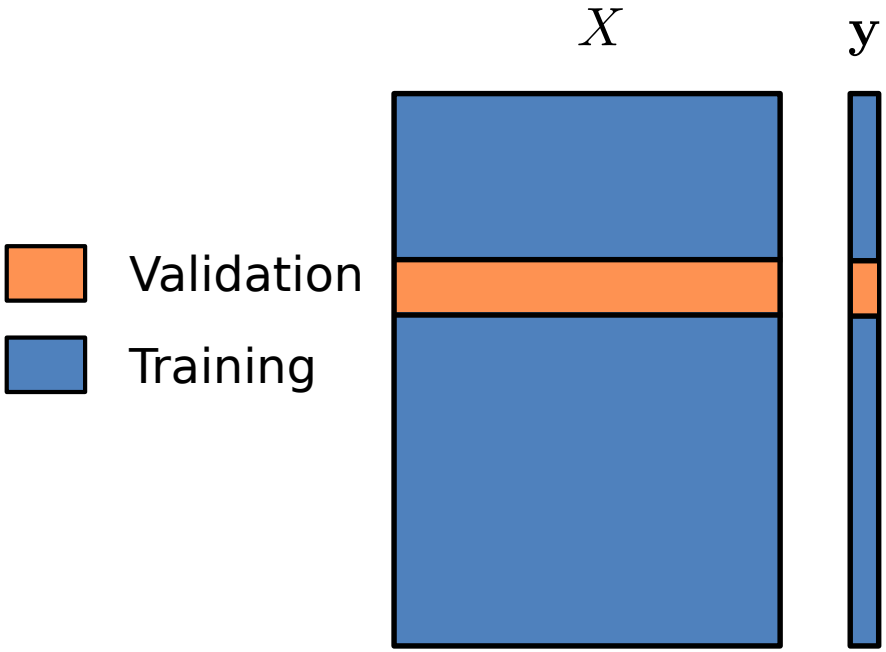


$k = 3$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

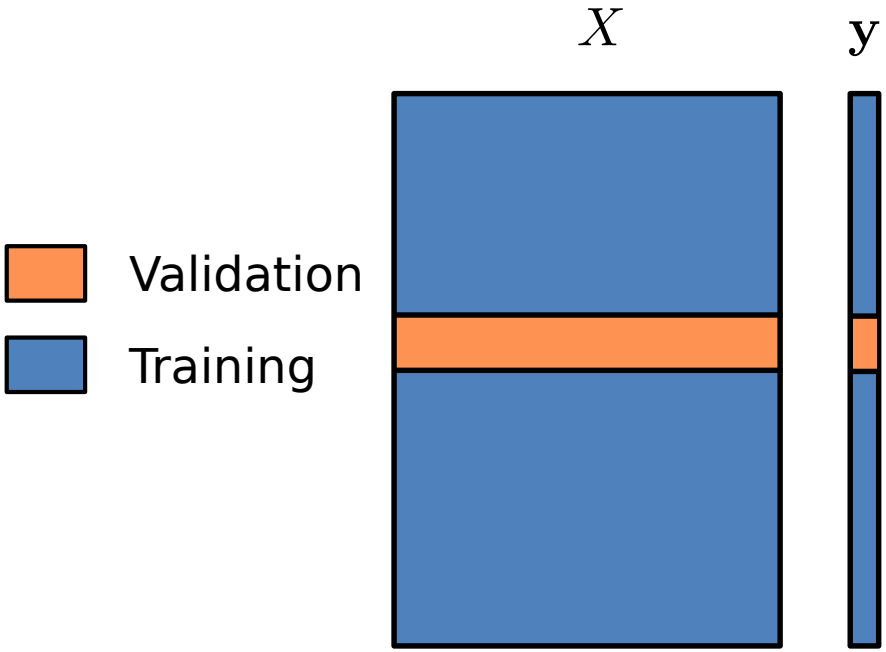


$k = 4$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

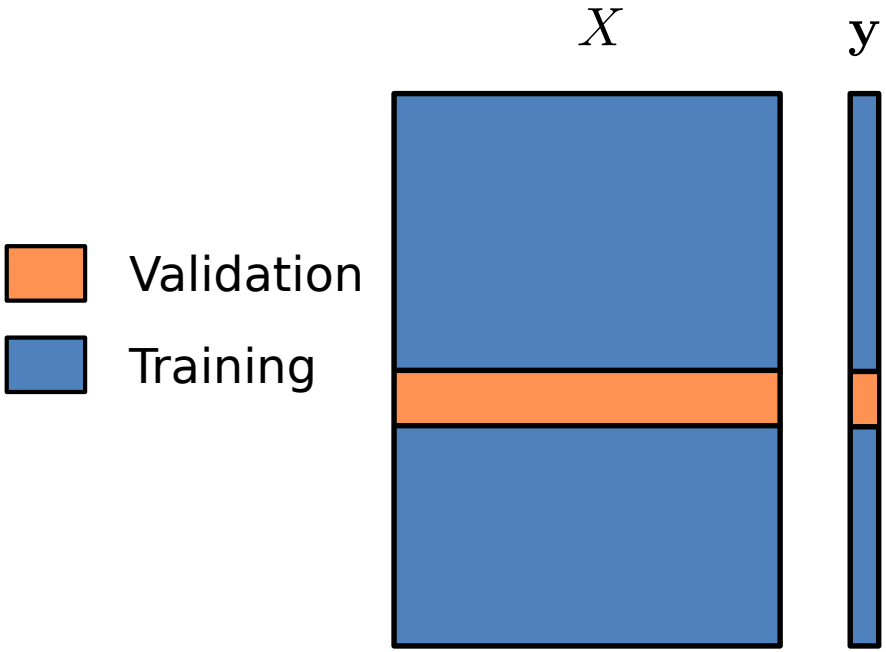


$k = 5$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

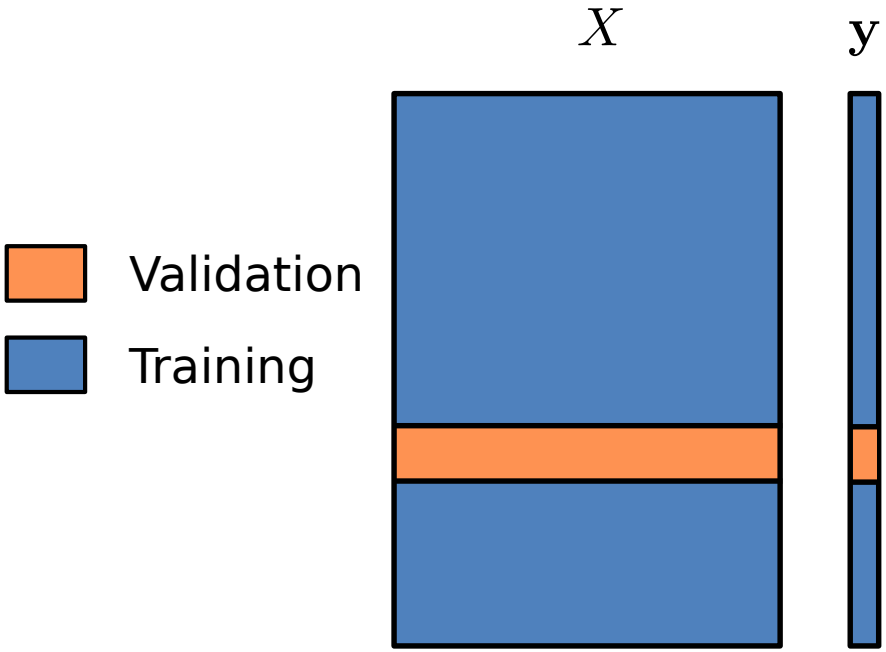


$k = 6$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

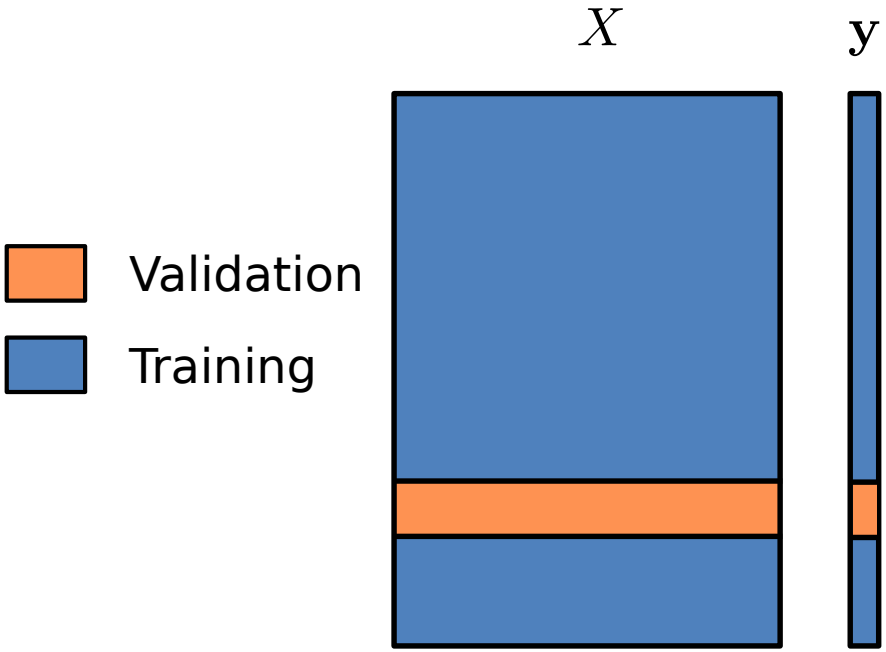


$k = 7$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

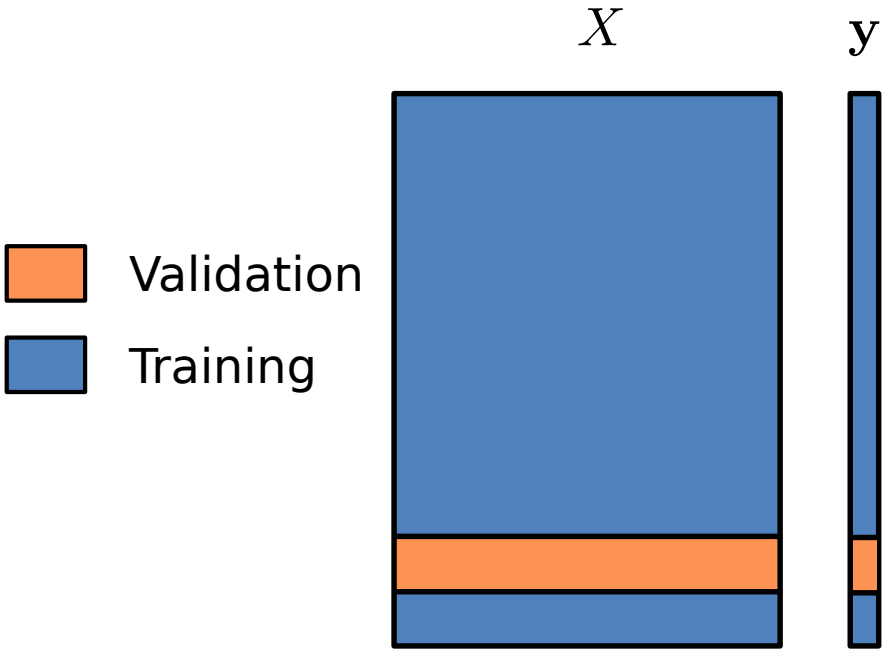


$k = 8$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

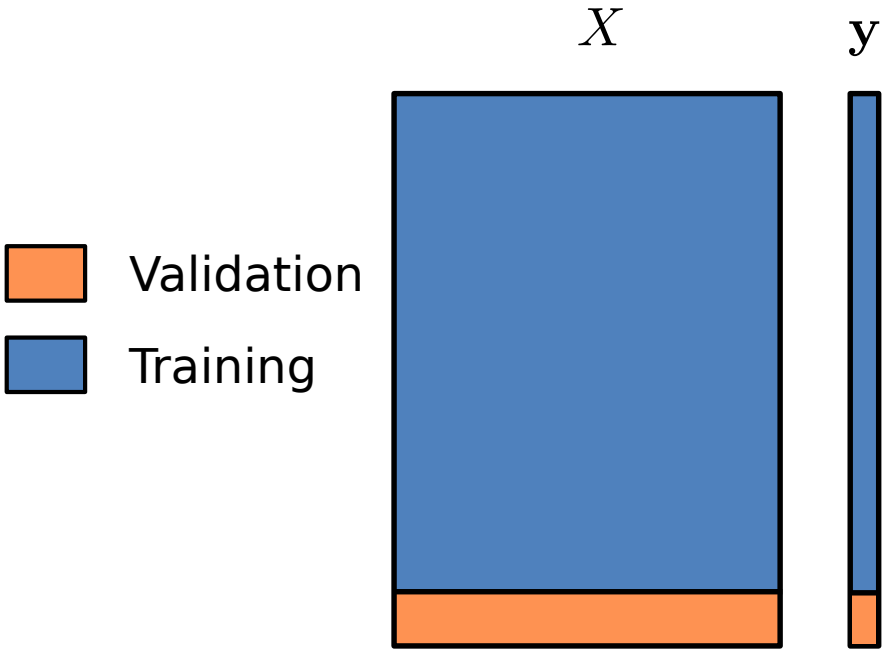


$k = 9$

1. Compute $\hat{\boldsymbol{\beta}}^{(\lambda_1)}, \ldots, \hat{\boldsymbol{\beta}}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

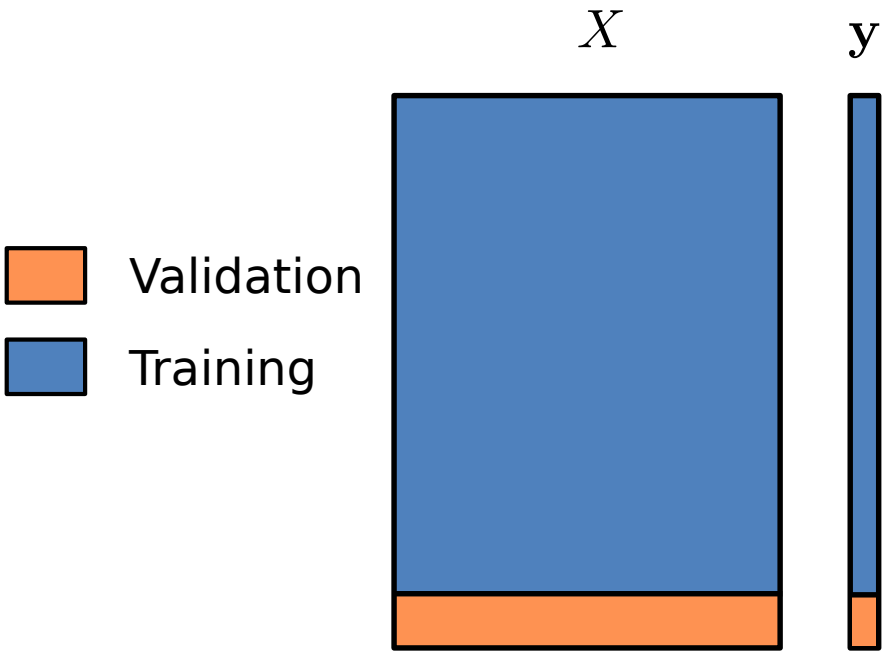


$k = 10$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



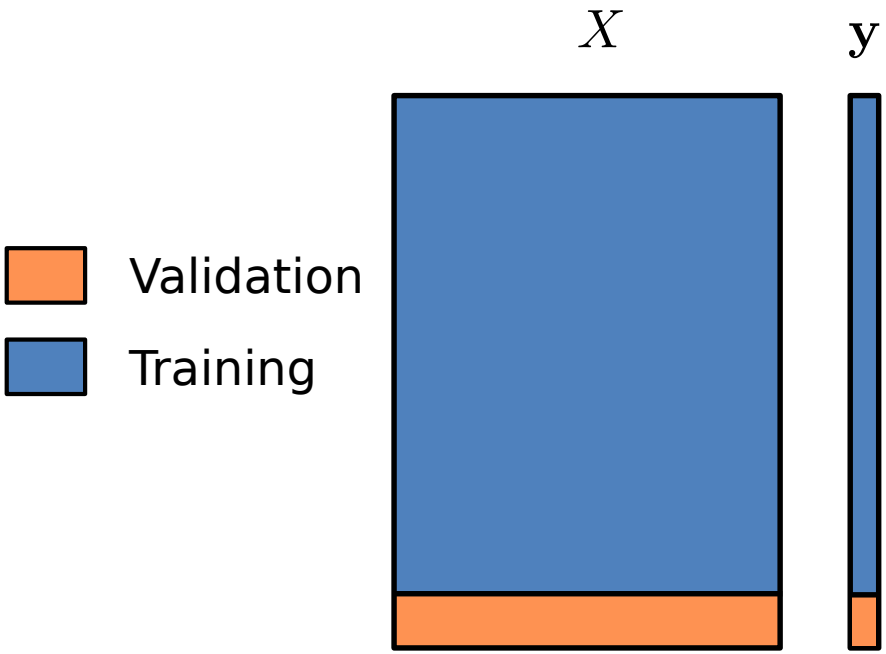
$k = 10$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_T^k$ on validation

Parameter choice: compute $\widehat{\mathrm{Error}}_1, \ldots, \widehat{\mathrm{Error}}_T$, average the errors and choose $\hat{i}^{\mathrm{CV}} \in [\![1, T]\!]$ achieving the smallest one

K-fold Cross-Validation ($K = 10$)

- Choose a grid of T λ's to test: $\lambda_1, \ldots, \lambda_T$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



$k = 10$

1. Compute $\hat{\boldsymbol{\beta}}^{(\lambda_1)}, \ldots, \hat{\boldsymbol{\beta}}^{(\lambda_T)}$ on train
2. Evaluate $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_T^k$ on validation

Parameter choice: compute $\widehat{\mathrm{Error}}_1, \ldots, \widehat{\mathrm{Error}}_T$, average the errors and choose $\hat{i}^{\mathrm{CV}} \in [\![1, T]\!]$ achieving the smallest one

Re-calibration: compute $\hat{\boldsymbol{\beta}}^{(\lambda_{\hat{i}^{\mathrm{CV}}})}$ this time over the whole sample

`sklearn` example

Common strategy: compute solutions over a grid, *i.e.*, get $\hat{\beta}^{(\lambda_1)}, \dots, \hat{\beta}^{(\lambda_T)}$, with $\lambda_1 > \cdots > \lambda_T$ for many T's, then pick the "best" one by cross-validation

Standard grid (R-glmnet / Python-sklearn) : geometric with $\lambda_1 = \|X^\top \mathbf{y}\|_\infty / n$, $\lambda_T = \lambda_1 / 1000$, $T = 100$
also motivated by optimization considerations,[12] K-fold cross-validation ($K = 10$, `glmnet`; $K = 5$, `sklearn`)

[12] E. Ndiaye et al. "Safe Grid Search with Optimal Complexity". In: *ICML*. vol. 97. 2019, pp. 4771–4780.

Outline

Convex optimization problem

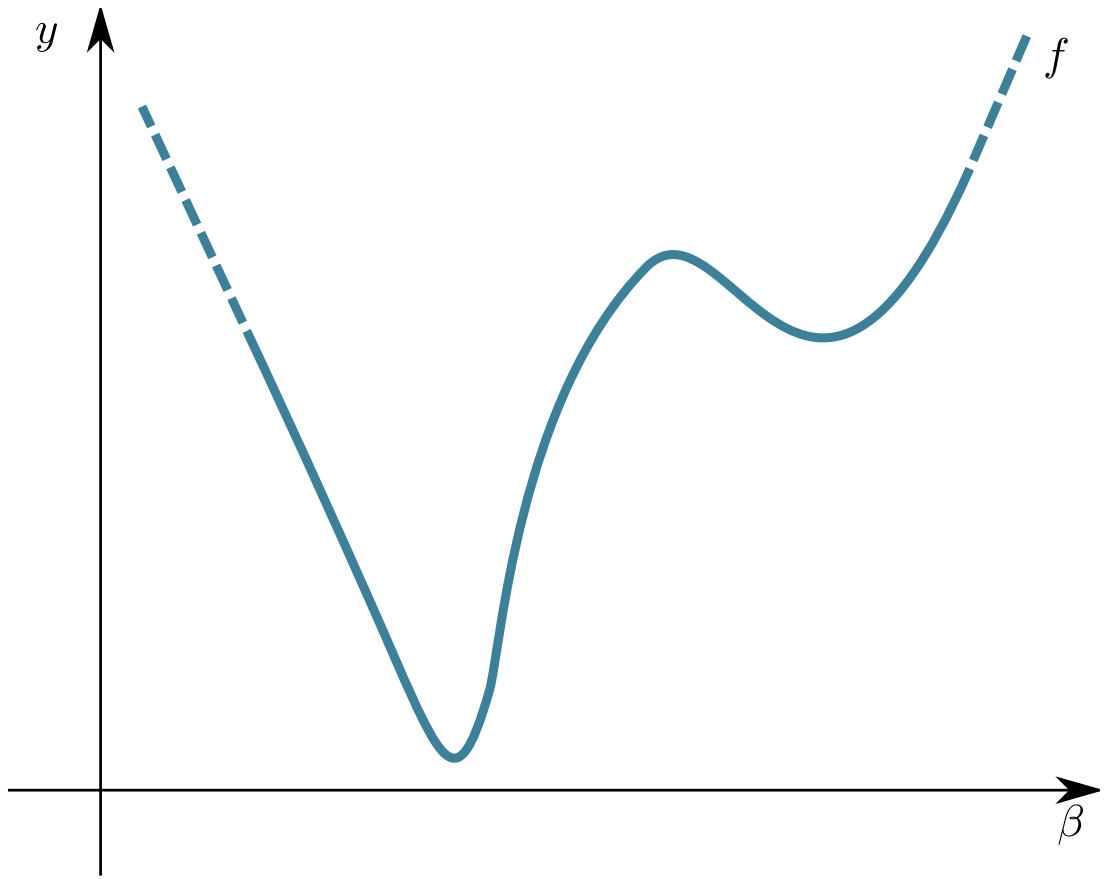
Problem formulation: $\boxed{\min_{\beta \in \mathbb{R}^p} f(\beta)}$

Questions:

- How to solve the Lasso problem?
- How to take into account structure / no structure?
- How to take into account large p?
- How to take into account expected sparsity of the problem?

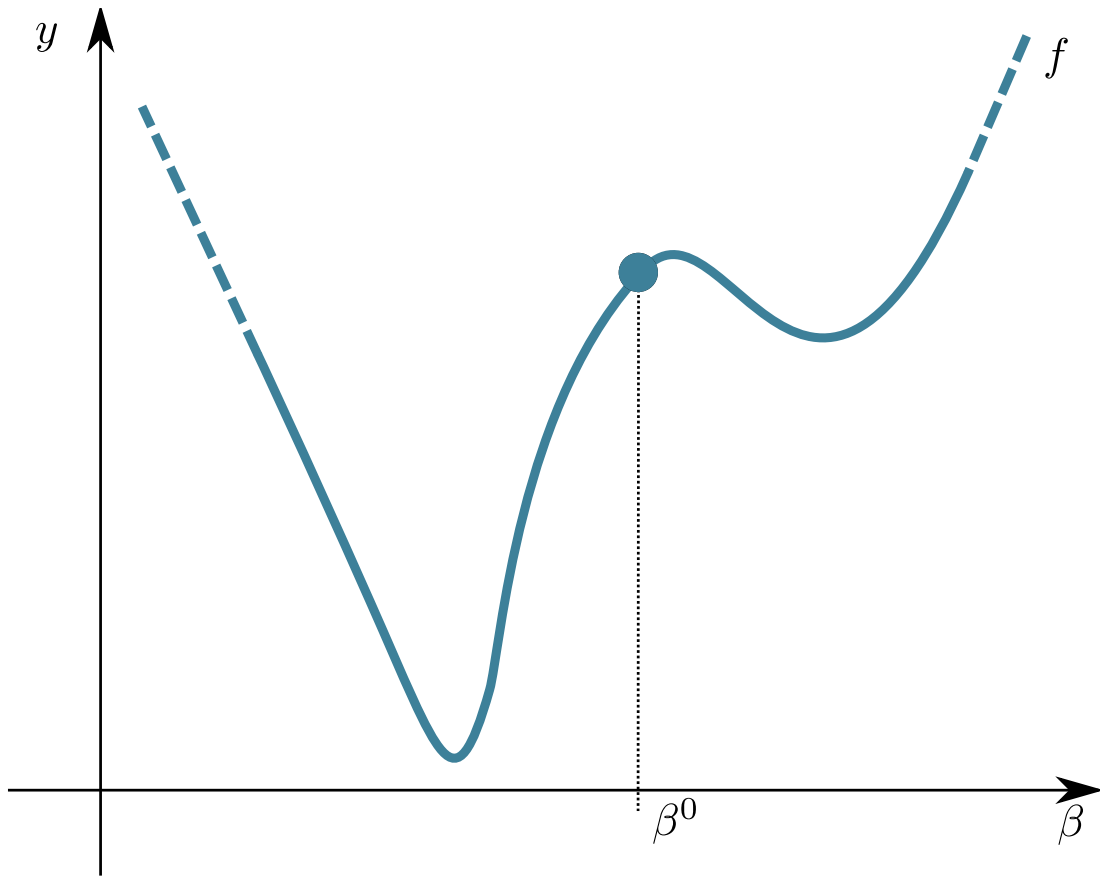
Outline

Majorization / Minimization: visually



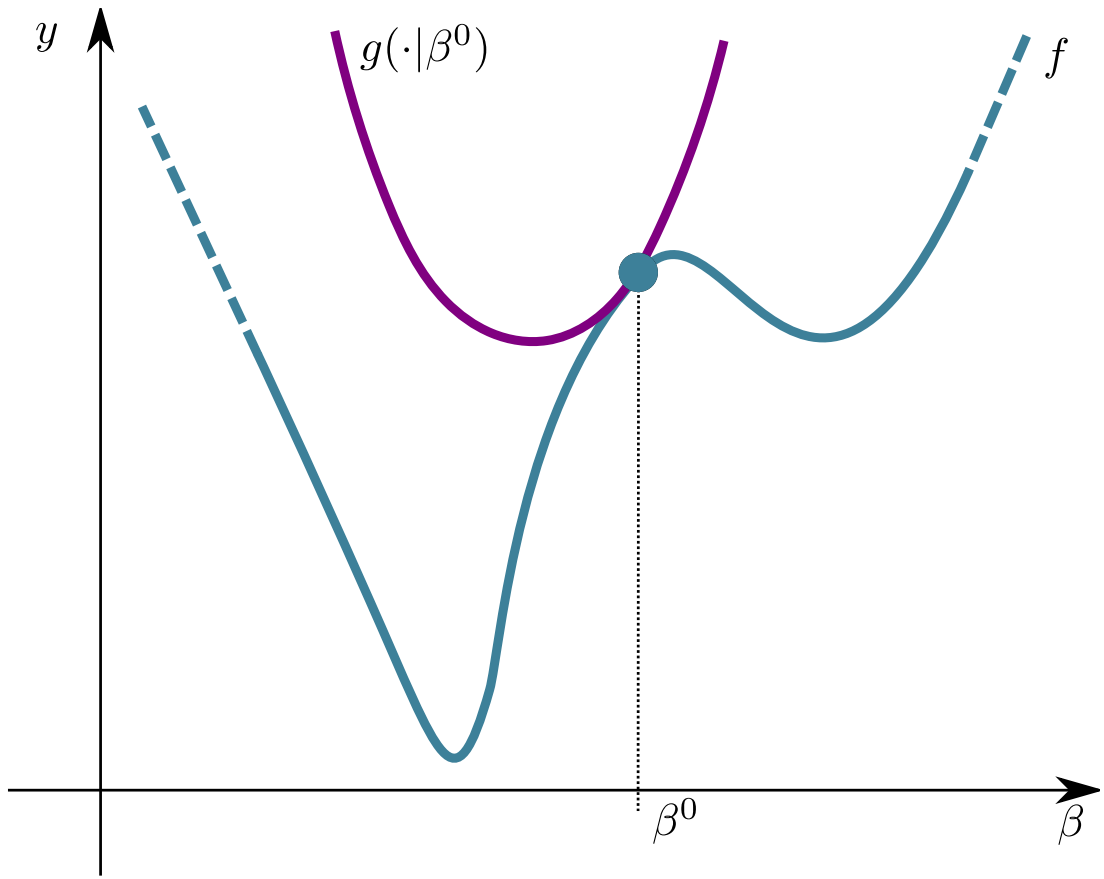
Original function

Majorization / Minimization: visually



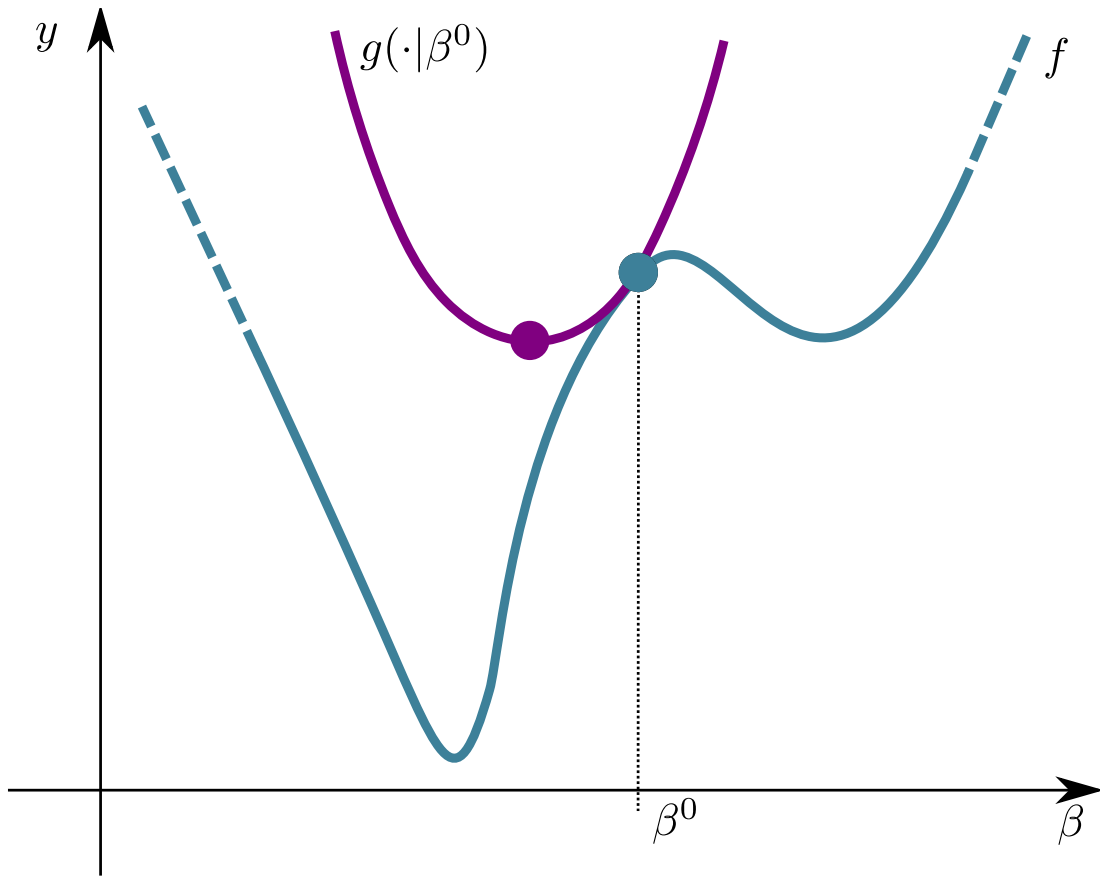
Initialize

Majorization / Minimization: visually



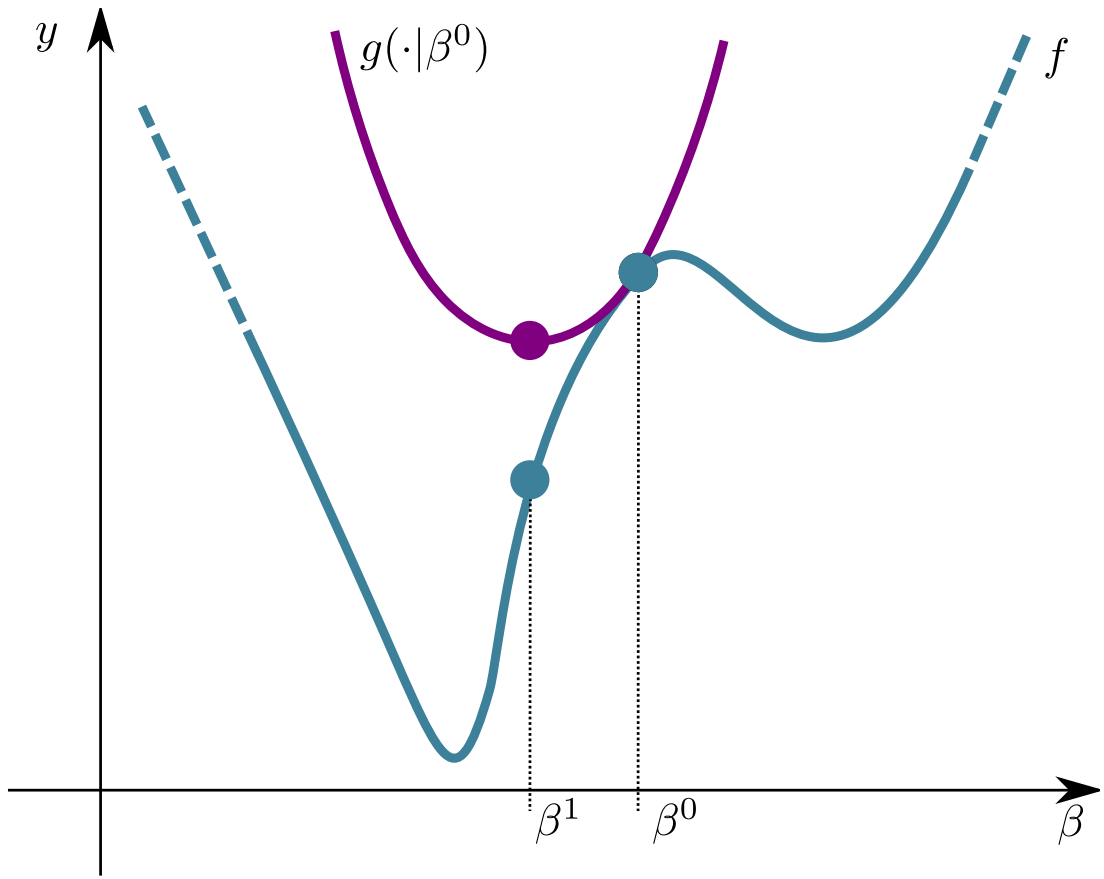
Majorize

Majorization / Minimization: visually



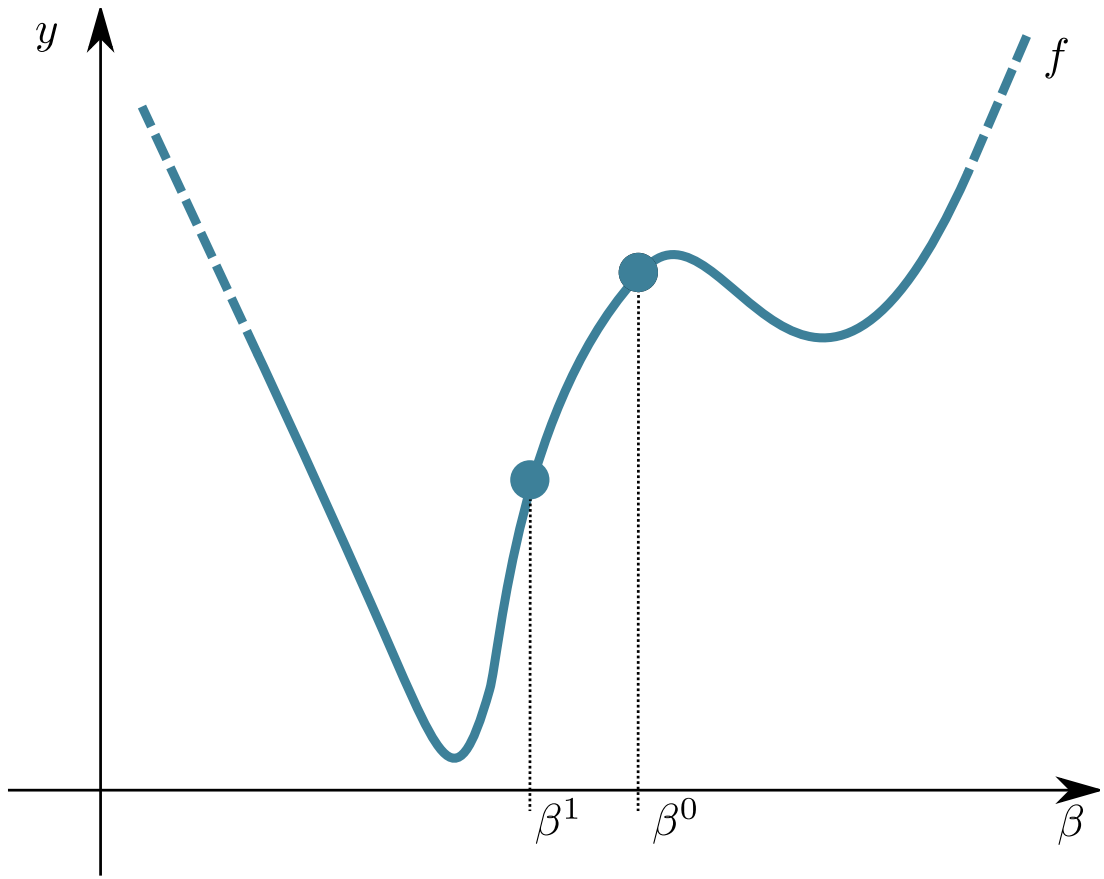
Minimize

Majorization / Minimization: visually



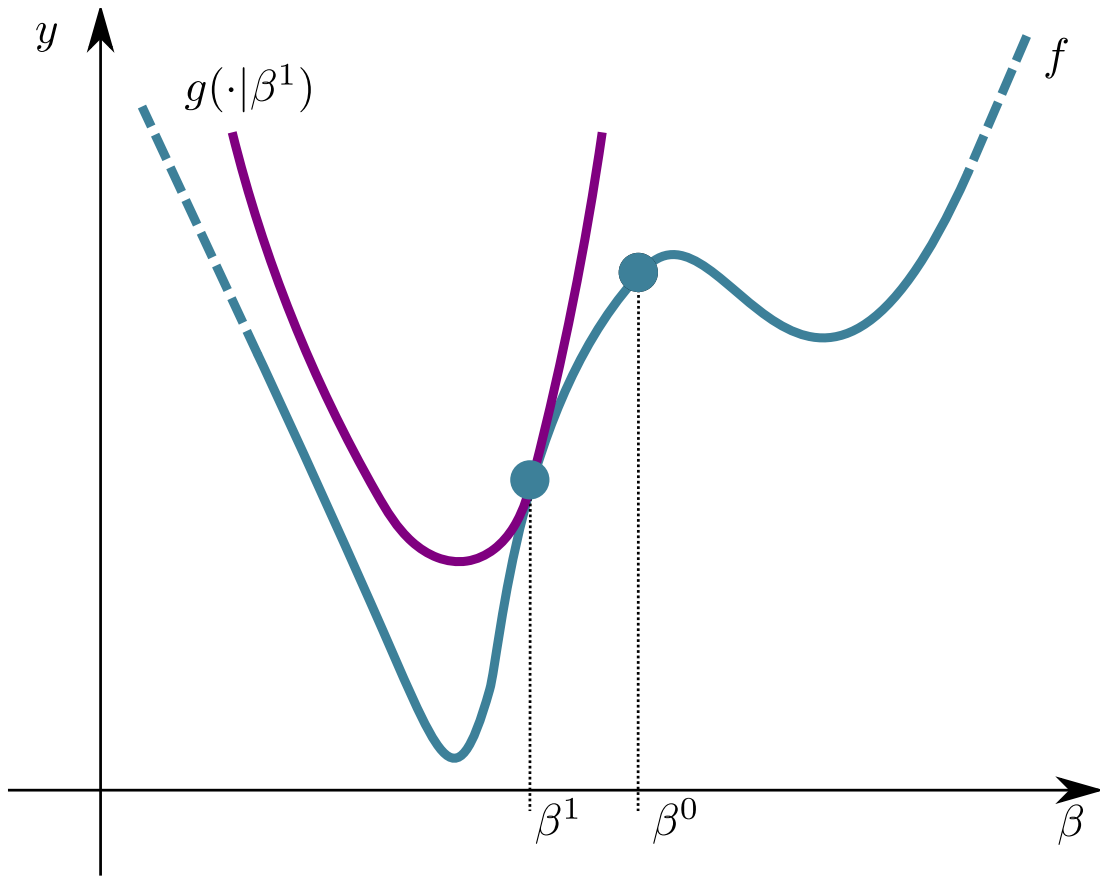
Update

Majorization / Minimization: visually



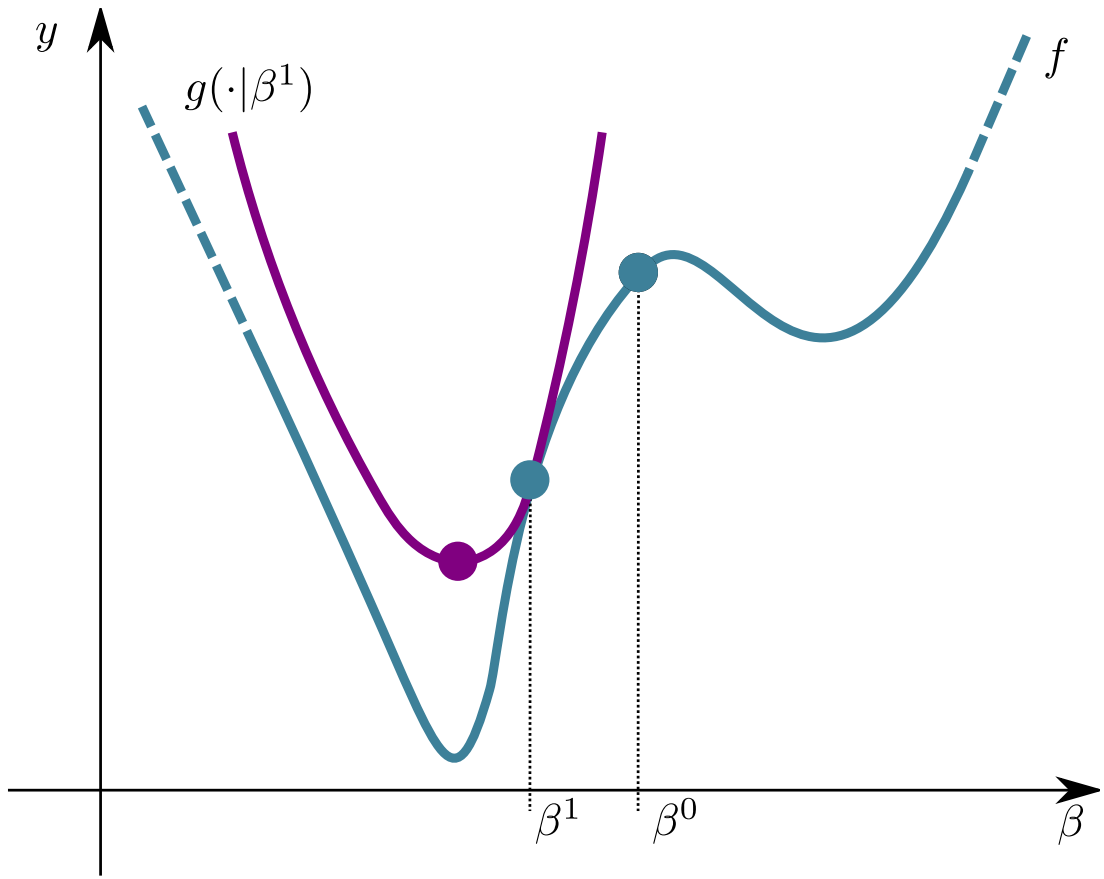
Update

Majorization / Minimization: visually



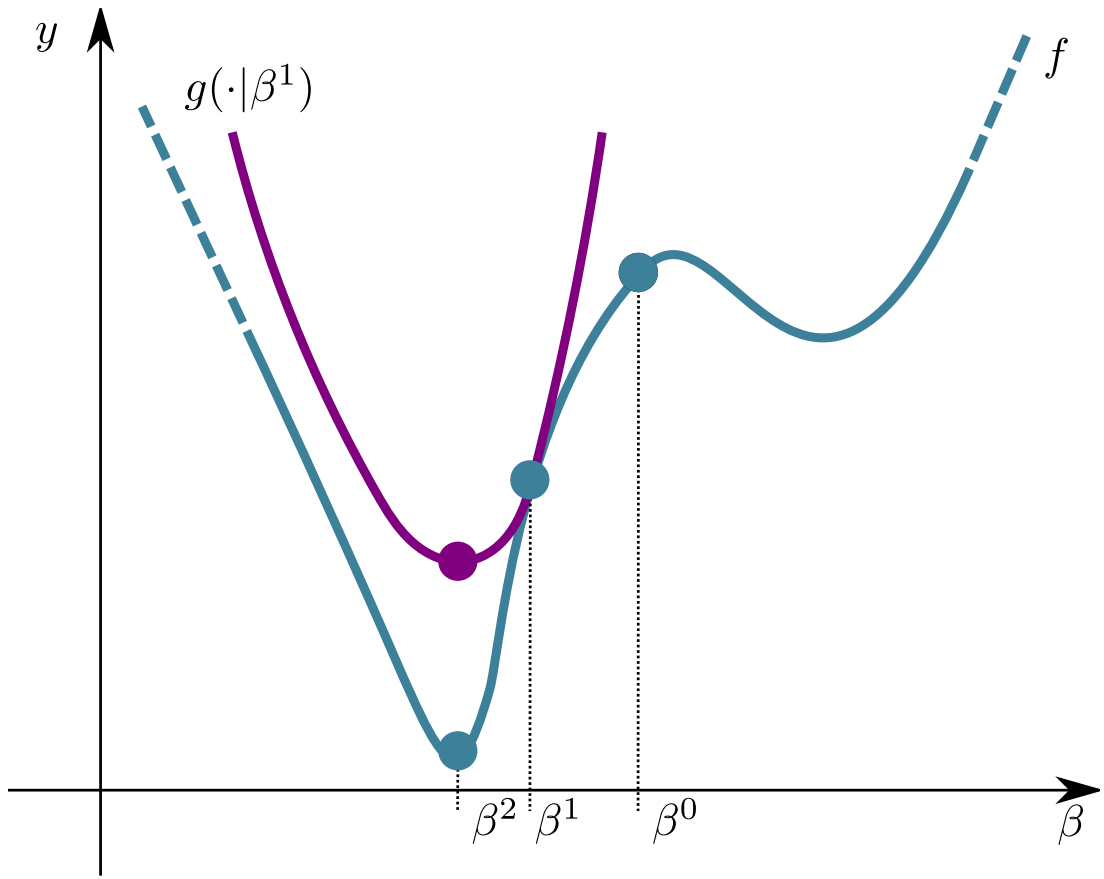
Majorize

Majorization / Minimization: visually



Minimize

Majorization / Minimization: visually



Update

Majorization / Minimization: formally

Objective: find a minimizer of a function f

Tool: at each point β^t proceed as follows:

- Provide a "**majorization**" function $\beta \mapsto g(\beta|\beta^t)$ satisfying:

$$\begin{cases} f(\beta) \leq g(\beta|\beta^t), \forall \beta & : \quad \text{domination / upper bound} \\ f(\beta^t) = g(\beta^t|\beta^t) & : \quad \text{tangency / tightness at } \beta^t \end{cases}$$

- **Minimize** the upper bound and obtain

$$\beta^{t+1} \in \underset{\beta \in \mathbb{R}^p}{\arg\min}\, g(\beta|\beta^t)$$

Majorization / Minimization: formally

Objective: find a minimizer of a function f

Tool: at each point $\boldsymbol{\beta}^t$ proceed as follows:

- Provide a "**majorization**" function $\boldsymbol{\beta} \mapsto g(\boldsymbol{\beta}|\boldsymbol{\beta}^t)$ satisfying:

$$\begin{cases} f(\boldsymbol{\beta}) \leq g(\boldsymbol{\beta}|\boldsymbol{\beta}^t), \forall \boldsymbol{\beta} & : \quad \text{domination / upper bound} \\ f(\boldsymbol{\beta}^t) = g(\boldsymbol{\beta}^t|\boldsymbol{\beta}^t) & : \quad \text{tangency / tightness at } \boldsymbol{\beta}^t \end{cases}$$

- **Minimize** the upper bound and obtain

$$\boldsymbol{\beta}^{t+1} \in \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min}\, g(\boldsymbol{\beta}|\boldsymbol{\beta}^t)$$

Rem: we say that $g(\cdot|\boldsymbol{\beta}^t)$ is a surrogate of f at $\boldsymbol{\beta}^t$

Majorization / Minimization: formally

Objective: find a minimizer of a function f

Tool: at each point $\boldsymbol{\beta}^t$ proceed as follows:

- Provide a "**majorization**" function $\boldsymbol{\beta} \mapsto g(\boldsymbol{\beta}|\boldsymbol{\beta}^t)$ satisfying:

$$\begin{cases} f(\boldsymbol{\beta}) \leq g(\boldsymbol{\beta}|\boldsymbol{\beta}^t), \forall \boldsymbol{\beta} & : \quad \text{domination / upper bound} \\ f(\boldsymbol{\beta}^t) = g(\boldsymbol{\beta}^t|\boldsymbol{\beta}^t) & : \quad \text{tangency / tightness at } \boldsymbol{\beta}^t \end{cases}$$

- **Minimize** the upper bound and obtain

$$\boldsymbol{\beta}^{t+1} \in \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min}\, g(\boldsymbol{\beta}|\boldsymbol{\beta}^t)$$

Rem: we say that $g(\cdot|\boldsymbol{\beta}^t)$ is a surrogate of f at $\boldsymbol{\beta}^t$

Majorization / Minimization algorithm

Algorithm: MAJORIZATION / MINIMIZATION

input : max. iterations $t_{\max}$, stopping criterion ϵ

init : $\boldsymbol{\beta}^0$

for $0 \leq t \leq t_{\max -1}$ **do**

  Break if stopping criterion smaller than ϵ

  Find a majorization function: $g(\cdot|\boldsymbol{\beta}^t)$

  Minimize it: $\boldsymbol{\beta}^{t+1} \leftarrow \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min}\, g(\boldsymbol{\beta}|\boldsymbol{\beta}^t)$

return $\boldsymbol{\beta}^{t_{\max}}$

Convergence property[13]

Theorem

The maximization/minimization algorithm is a descent method:

$$\forall t \geq 0, \quad f(\boldsymbol{\beta}^{t+1}) \leq f(\boldsymbol{\beta}^{t})$$

Hence, provided that f is lower bounded the algorithm converges.

[13] K. Lange. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.

Convergence property[13]

Theorem

The maximization/minimization algorithm is a descent method:

$$\forall t \geq 0, \quad f(\boldsymbol{\beta}^{t+1}) \leq f(\boldsymbol{\beta}^{t})$$

Hence, provided that f is lower bounded the algorithm converges.

Proof:

[13] K. Lange. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.

Convergence property[13]

Theorem

The maximization/minimization algorithm is a descent method:

$$\forall t \geq 0, \quad f(\boldsymbol{\beta}^{t+1}) \leq f(\boldsymbol{\beta}^{t})$$

Hence, provided that f is lower bounded the algorithm converges.

Proof:

$$f(\boldsymbol{\beta}^{t+1}) \leq g(\boldsymbol{\beta}^{t+1}|\boldsymbol{\beta}^{t}) \qquad \text{(Majorization at } \boldsymbol{\beta}^{t}\text{)}$$

[13] K. Lange. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.

Convergence property[13]

Theorem

The maximization/minimization algorithm is a descent method:

$$\forall t \geq 0, \quad f(\boldsymbol{\beta}^{t+1}) \leq f(\boldsymbol{\beta}^{t})$$

Hence, provided that f is lower bounded the algorithm converges.

Proof:

$$\begin{aligned} f(\boldsymbol{\beta}^{t+1}) &\leq g(\boldsymbol{\beta}^{t+1}|\boldsymbol{\beta}^{t}) && \text{(Majorization at } \boldsymbol{\beta}^{t}\text{)} \\ &\leq g(\boldsymbol{\beta}^{t}|\boldsymbol{\beta}^{t}) && \text{(Minimization definition of } \boldsymbol{\beta}^{t+1}\text{)} \end{aligned}$$

[13] K. Lange. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.

Convergence property[13]

Theorem

The maximization/minimization algorithm is a descent method:

$$\forall t \geq 0, \quad f(\boldsymbol{\beta}^{t+1}) \leq f(\boldsymbol{\beta}^{t})$$

Hence, provided that f is lower bounded the algorithm converges.

Proof:

$$\begin{aligned} f(\boldsymbol{\beta}^{t+1}) &\leq g(\boldsymbol{\beta}^{t+1}|\boldsymbol{\beta}^{t}) && \text{(Majorization at } \boldsymbol{\beta}^{t}) \\ &\leq g(\boldsymbol{\beta}^{t}|\boldsymbol{\beta}^{t}) && \text{(Minimization definition of } \boldsymbol{\beta}^{t+1}) \\ &= f(\boldsymbol{\beta}^{t}) && \text{(tightness at } \boldsymbol{\beta}^{t}) \end{aligned}$$

[13] K. Lange. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.

Gradient descent revisited

Optimization problem: $\boxed{\min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta})}$

Properties: f is convex with gradient L-Lipschitz

$$\forall (\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\boldsymbol{\beta}) - \nabla f(\boldsymbol{\beta}')\| \leq L \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|$$

Gradient descent revisited

Optimization problem: $\boxed{\min_{\beta \in \mathbb{R}^p} f(\beta)}$

Properties: f is convex with gradient L-Lipschitz

$$\forall (\beta, \beta') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\beta) - \nabla f(\beta')\| \leq L \|\beta - \beta'\|$$

(quadratic) Surrogate: (proof in Appendix)

$$g(\beta | \beta^t) = f(\beta^t) + \langle \nabla f(\beta^t), \beta - \beta^t \rangle + \frac{L}{2} \|\beta^t - \beta\|^2$$

Gradient descent revisited

Optimization problem: $\boxed{\min_{\beta \in \mathbb{R}^p} f(\beta)}$

Properties: f is convex with gradient L-Lipschitz

$$\forall (\beta, \beta') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\beta) - \nabla f(\beta')\| \leq L\|\beta - \beta'\|$$

(quadratic) Surrogate: (proof in Appendix)

$$g(\beta|\beta^t) = f(\beta^t) + \langle \nabla f(\beta^t), \beta - \beta^t \rangle + \frac{L}{2}\|\beta^t - \beta\|^2$$

Update rule : $\boxed{\beta^{t+1} = \beta^t - \frac{1}{L}\nabla f(\beta^t)}$

Gradient descent revisited

Optimization problem: $\boxed{\min_{\beta \in \mathbb{R}^p} f(\beta)}$

Properties: f is convex with gradient L-Lipschitz

$$\forall (\beta, \beta') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\beta) - \nabla f(\beta')\| \leq L \|\beta - \beta'\|$$

(quadratic) Surrogate: (proof in Appendix)

$$g(\beta | \beta^t) = f(\beta^t) + \langle \nabla f(\beta^t), \beta - \beta^t \rangle + \frac{L}{2} \|\beta^t - \beta\|^2$$

Update rule : $\boxed{\beta^{t+1} = \beta^t - \frac{1}{L} \nabla f(\beta^t)}$

Rem: $\alpha \leq 1/L$ also works as a step size

Outline

Proximal gradient descent: non-smooth case

Optimization problem: $\boxed{\min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta}) + \psi(\boldsymbol{\beta})}$

Properties: f convex, gradient L-Lipschitz; ψ but non necessarily smooth (can have kinks)

Example: $f(\boldsymbol{\beta}) = \frac{1}{2n} \|\mathbf{y} - X\boldsymbol{\beta}\|^2, \psi(\boldsymbol{\beta}) = \lambda \|\boldsymbol{\beta}\|_1$

Proximal operators / algorithms

Properties: f convex, gradient L-Lipschitz; ψ convex s.t. prox_ψ (the **proximal** operator[14] of ψ) has a closed-form, where

$$\boxed{\mathrm{prox}_\psi\left(\boldsymbol{\beta}^0\right) := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \tfrac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\|^2 + \psi(\boldsymbol{\beta})}$$

[14] J.-J. Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *C. R. Acad. Sci. Paris* 255 (1962), pp. 2897–2899.

[15] S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[16] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011, pp. xvi+468.

Proximal operators / algorithms

Properties: f convex, gradient L-Lipschitz; ψ convex s.t. prox_ψ (the **proximal** operator[14] of ψ) has a closed-form, where

$$\boxed{\mathrm{prox}_\psi\left(\boldsymbol{\beta}^0\right) := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \tfrac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\|^2 + \psi(\boldsymbol{\beta})}$$

Surrogate: $g(\boldsymbol{\beta}|\boldsymbol{\beta}^t) = f(\boldsymbol{\beta}^t) + \langle \nabla f(\boldsymbol{\beta}^t), \boldsymbol{\beta}-\boldsymbol{\beta}^t\rangle + \frac{L\|\boldsymbol{\beta}^t-\boldsymbol{\beta}\|^2}{2} + \psi(\boldsymbol{\beta})$

---

[14] J.-J. Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *C. R. Acad. Sci. Paris* 255 (1962), pp. 2897–2899.

[15] S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[16] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011, pp. xvi+468.

Proximal operators / algorithms

Properties: f convex, gradient L-Lipschitz; ψ convex s.t. prox_ψ (the **proximal** operator[14] of ψ) has a closed-form, where

$$\boxed{\text{prox}_\psi\left(\boldsymbol{\beta}^0\right) := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \tfrac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\|^2 + \psi(\boldsymbol{\beta})}$$

Surrogate: $g(\boldsymbol{\beta}|\boldsymbol{\beta}^t) = f(\boldsymbol{\beta}^t) + \langle \nabla f(\boldsymbol{\beta}^t), \boldsymbol{\beta}-\boldsymbol{\beta}^t\rangle + \frac{L\|\boldsymbol{\beta}^t-\boldsymbol{\beta}\|^2}{2} + \psi(\boldsymbol{\beta})$

Update rule : $$\boxed{\boldsymbol{\beta}^{t+1} = \text{prox}_{\frac{\psi}{L}}\left(\boldsymbol{\beta}^t - \tfrac{1}{L}\nabla f(\boldsymbol{\beta}^t)\right)}$$

---

(14) J.-J. Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *C. R. Acad. Sci. Paris* 255 (1962), pp. 2897–2899.

(15) S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

(16) H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011, pp. xvi+468.

Proximal operators / algorithms

Properties: f convex, gradient L-Lipschitz; ψ convex s.t. prox_ψ (the **proximal** operator[14] of ψ) has a closed-form, where

$$\boxed{\text{prox}_\psi\left(\boldsymbol{\beta}^0\right) := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \tfrac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\|^2 + \psi(\boldsymbol{\beta})}$$

Surrogate: $g(\boldsymbol{\beta}|\boldsymbol{\beta}^t) = f(\boldsymbol{\beta}^t) + \langle \nabla f(\boldsymbol{\beta}^t), \boldsymbol{\beta}-\boldsymbol{\beta}^t\rangle + \frac{L\|\boldsymbol{\beta}^t-\boldsymbol{\beta}\|^2}{2} + \psi(\boldsymbol{\beta})$

Update rule : $$\boxed{\boldsymbol{\beta}^{t+1} = \text{prox}_{\frac{\psi}{L}}\left(\boldsymbol{\beta}^t - \tfrac{1}{L}\nabla f(\boldsymbol{\beta}^t)\right)}$$

More details on prox properties:

- Prox algorithms recipes[15]
- Mathematical theory/analysis[16]

[14] J.-J. Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *C. R. Acad. Sci. Paris* 255 (1962), pp. 2897–2899.

[15] S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[16] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011, pp. xvi+468.

Examples of prox operators

$$\boxed{\operatorname{prox}_{\psi}(\boldsymbol{\beta}^0) := \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left(\tfrac{1}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\|^2 + \psi(\boldsymbol{\beta}) \right)}$$

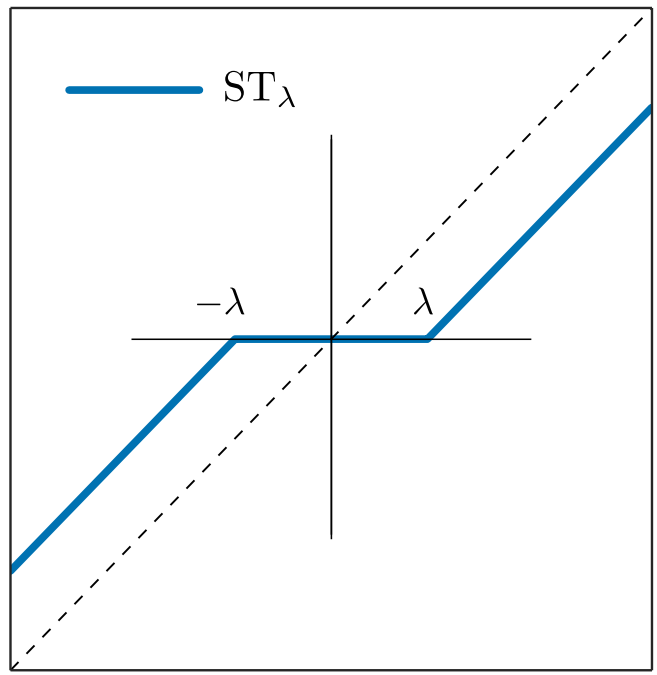
- **Null function**: $\psi = 0$, then $\operatorname{prox}_{\psi} = \mathrm{Id}$
- **Indicator function**: $\psi = \iota_C$ for a closed convex set $C \subset \mathbb{R}^p$, then $\operatorname{prox}_{\psi} = \pi_C$, projection over the set C
- **Soft-Thresholding**: $\psi(\cdot) = \lambda|\cdot|$, then $\operatorname{prox}_{\psi}(\boldsymbol{\beta}^0) = \mathrm{ST}_{\lambda}(\boldsymbol{\beta}^0) = \operatorname{sign}(\boldsymbol{\beta}^0)(|\boldsymbol{\beta}^0| - \lambda)_+$
- **Soft-Thresholding (vector)**: $\psi(\cdot) = \lambda\|\cdot\|_1$, then $\operatorname{prox}_{\psi}(\boldsymbol{\beta}^0) = (\mathrm{ST}_{\lambda}(\boldsymbol{\beta}^0_1), \ldots, \mathrm{ST}_{\lambda}(\boldsymbol{\beta}^0_p))^\top$

Soft-Thresholding

Closed form solution for 1D-problem ($p = 1$) : **Soft-Thresholding**

$$\begin{aligned}\mathrm{ST}_\lambda(\beta^0) &:= \arg\min_{\beta\in\mathbb{R}} \left(\frac{(\beta^0 - \beta)^2}{2} + \lambda|\beta| \right) \\ &= \mathrm{sign}(\beta^0)(|\beta^0| - \lambda)_+\end{aligned}$$

with $(\cdot)_+ := \max(0, \cdot)$



Forward-Backward
Iterative Soft Thresholding (ISTA)

Optimization problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta}) + \psi(\boldsymbol{\beta})$$

Extension of gradient descent for composite functions:

General Forward-Backward

Choose step size value: α
Initialization: $\boldsymbol{\beta} = 0 \in \mathbb{R}^p$
While not converged
$\boldsymbol{\beta} \leftarrow \text{prox}_{\alpha\psi}\left(\boldsymbol{\beta} - \alpha \nabla f(\boldsymbol{\beta})\right)$

Forward-Backward
Iterative Soft Thresholding (ISTA)

Optimization problem:

$$\min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$$

Extension of gradient descent for composite functions:

General Forward-Backward	Iterative Soft-thresholding (ISTA)
Choose step size value: α Initialization: $\beta = 0 \in \mathbb{R}^p$ While not converged $\beta \leftarrow \text{prox}_{\alpha\psi}\left(\beta - \alpha \nabla f(\beta)\right)$	Choose step size value: α Initialization: $\beta = 0 \in \mathbb{R}^p$ While not converged $\beta \leftarrow \text{ST}_{n\alpha\lambda}\left(\beta + \frac{\alpha}{n} X^\top (y - X\beta)\right)$
	$f(\beta) = \frac{1}{2n} \left\|X\beta - y\right\|^2,$ $\psi(\beta) = \lambda \left\|\beta\right\|_1$

Forward-Backward / Iterative Soft Thresholding (ISTA) (II)

- Requires α to be tuned: often set $\alpha = 1/L = 1/\mu_{\max}(\frac{X^\top X}{n})$ ($\mu_{\max}(\frac{X^\top X}{n})$ spectral radius of $\frac{X^\top X}{n}$), or by line-search
- Acceleration : Fast Iterative Soft Thresholding Algorithm (FISTA)[17],[18] (momentum[19])

[17] Y. Nesterov. "A method for solving a convex programming problem with rate of convergence $O(1/k^2)$". In: *Soviet Math. Doklady* 269.3 (1983), pp. 543–547.

[18] A. Beck and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM J. Imaging Sci.* 2.1 (2009), pp. 183–202.

[19] https://distill.pub/2017/momentum/

The Lasso: popular algorithms

Commonly used algorithms for solving this **convex** program:

- Homotopy method - LARS:
  efficient for small p Osborne *et al.* (2000), Efron *et al.* (2004)
  and to get full path (*i.e.*, the full $\lambda \mapsto \hat{\beta}^{(\lambda)}$)
  Limitation: do not generalize to other data-fitting term, potentially too many kinks Mairal and Yu (2012) (up to 3^p)

- (F)ISTA, Forward - Backward, proximal algorithm:
  useful when $r \to X^\top r$ is cheap to compute (*e.g.*, FFT, Fast Wavelet Transform, etc.) Beck and Teboulle (2009)
  Limitation: unstructured X in statistics / machine learning

The Lasso: popular algorithms

Commonly used algorithms for solving this **convex** program:

- Homotopy method - LARS:
  efficient for small p Osborne *et al.* (2000), Efron *et al.* (2004)
  and to get full path (*i.e.*, the full $\lambda \mapsto \hat{\beta}^{(\lambda)}$)
  Limitation: do not generalize to other data-fitting term, potentially too many kinks Mairal and Yu (2012) (up to 3^p)

- (F)ISTA, Forward - Backward, proximal algorithm:
  useful when $r \to X^\top r$ is cheap to compute (*e.g.*, FFT, Fast Wavelet Transform, etc.) Beck and Teboulle (2009)
  Limitation: unstructured X in statistics / machine learning

- Coordinate descent:
  useful for large p and (unstructured) sparse matrix X, *e.g.*, for text encoding Friedman *et al.* (2007)
  Conclusion: standard approach in machine learning/statistics

The Lasso: popular algorithms

Commonly used algorithms for solving this **convex** program:

- Homotopy method - LARS:
  efficient for small p Osborne *et al.* (2000), Efron *et al.* (2004)
  and to get full path (*i.e.*, the full $\lambda \mapsto \hat{\beta}^{(\lambda)}$)
  Limitation: do not generalize to other data-fitting term, potentially too many kinks Mairal and Yu (2012) (up to 3^p)

- (F)ISTA, Forward - Backward, proximal algorithm:
  useful when $r \to X^\top r$ is cheap to compute (*e.g.*, FFT, Fast Wavelet Transform, etc.) Beck and Teboulle (2009)
  Limitation: unstructured X in statistics / machine learning

- Coordinate descent:
  useful for large p and (unstructured) sparse matrix X, *e.g.*, for text encoding Friedman *et al.* (2007)
  Conclusion: standard approach in machine learning/statistics

Outline

Coordinate Descent

Objective: $\min_{\boldsymbol{\beta} \in \mathbb{R}^p} F(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta}) + \psi(\boldsymbol{\beta})$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\boldsymbol{\beta}^{(k)} = 0 \in \mathbb{R}^p$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \ldots, K$ **do**

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \dots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \quad, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \ldots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \quad, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \ldots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \approx \arg\min_{\beta_2 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2 \quad, \beta_3^{(k-1)}, \ldots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \dots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \quad, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \approx \arg\min_{\beta_2 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2 \quad, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_3^{(k)} \approx \arg\min_{\beta_3 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \quad, \beta_3 \quad, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \dots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \quad, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \approx \arg\min_{\beta_2 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2 \quad, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_3^{(k)} \approx \arg\min_{\beta_3 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \quad, \beta_3 \quad, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$\vdots$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \dots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \ , \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \approx \arg\min_{\beta_2 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2 \ , \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_3^{(k)} \approx \arg\min_{\beta_3 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \ , \beta_3 \ , \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\vdots$$

$$\beta_p^{(k)} \approx \arg\min_{\beta_p \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \ , \beta_3^{(k)} \ , \dots, \beta_{p-1}^{(k)} \ , \beta_p \ )$$

Coordinate Descent

Objective: $\min_{\beta \in \mathbb{R}^p} F(\beta) = \min_{\beta \in \mathbb{R}^p} f(\beta) + \psi(\beta)$

Algorithm: Coordinate Descent

Input : F, epochs K (or passes over the data)

Init: $k = 0$ and $\beta^{(k)} = 0 \in \mathbb{R}^p$

for $k = 1, \dots, K$ **do**

$$\beta_1^{(k)} \approx \arg\min_{\beta_1 \in \mathbb{R}} F(\beta_1 \ , \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} \approx \arg\min_{\beta_2 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2 \ , \beta_3^{(k-1)}, \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\beta_3^{(k)} \approx \arg\min_{\beta_3 \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \ , \beta_3 \ , \dots, \beta_{p-1}^{(k-1)}, \beta_p^{(k-1)})$$

$$\vdots$$

$$\beta_p^{(k)} \approx \arg\min_{\beta_p \in \mathbb{R}} F(\beta_1^{(k)}, \beta_2^{(k)} \ , \beta_3^{(k)} \ , \dots, \beta_{p-1}^{(k)} \ , \beta_p \ )$$

Output : $\beta^{(K)}$

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$
- **Random**: *i.i.d.* uniformly with resampling

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$
- **Random**: *i.i.d.* uniformly with resampling
- **Shuffle**: uniform permutations

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$
- **Random**: *i.i.d.* uniformly with resampling
- **Shuffle**: uniform permutations

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$
- **Random**: *i.i.d.* uniformly with resampling
- **Shuffle**: uniform permutations

Rem: same idea used in linear solvers

Popular visit schemes

Need to visit coordinates "often" (or "greedily") for convergence

Most common ones:

- **Cyclic** (Gauss-Seidel): visit $1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$
- **Random**: *i.i.d.* uniformly with resampling
- **Shuffle**: uniform permutations

Rem: same idea used in linear solvers

Rem: coordinate-wise proximal gradient descent is also a Maximization/Minimization method

Motivation for coordinate descent

- useful when many features (p large)
- "block" strategy: update a block (or one coordinate at a time)
- convergence guarantees:

1. Smooth case[20]: $\arg\min_{\beta} f(\beta)$

   with f convex and gradient Lipschitz

2. Composite case[21] : $\arg\min_{\beta} f(\beta) + \psi(\beta)$

   f convex and gradient Lipschitz, and ψ convex **separable**:

   $$\psi(\beta) = \sum_{j=1}^{p} \psi_j(\beta_j)$$

---

[20] B. Martinet. "Brève communication. Régularisation d'inéquations variationnelles par approximations successives". In: *Revue française d'informatique et de recherche opérationnelle. Série rouge* 4.R3 (1970), pp. 154–158.

[21] P. Tseng. "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *J. Optim. Theory Appl.* 109.3 (2001), pp. 475–494.

Motivation for coordinate descent

- useful when many features (p large)
- "block" strategy: update a block (or one coordinate at a time)
- convergence guarantees:

1. Smooth case[20]: $\arg\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta})$

   with f convex and gradient Lipschitz

2. Composite case[21] : $\arg\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) + \psi(\boldsymbol{\beta})$

   f convex and gradient Lipschitz, and ψ convex **separable**:

   $$\psi(\boldsymbol{\beta}) = \sum_{j=1}^{p} \psi_j(\beta_j)$$

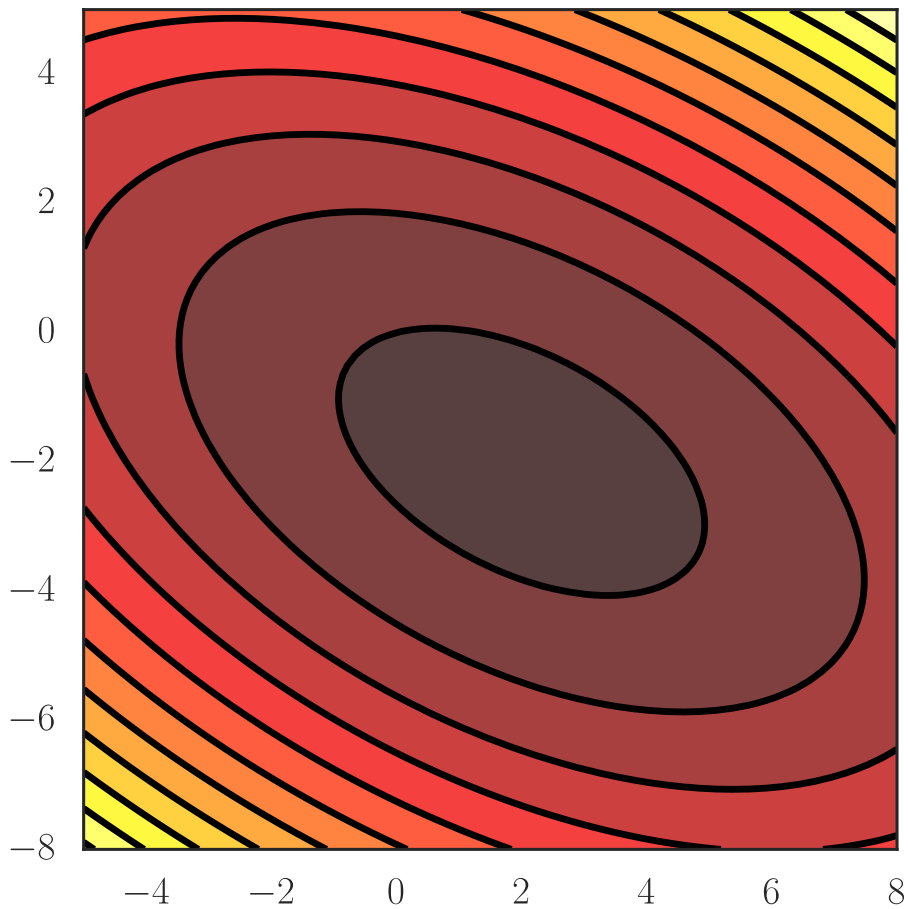
---

[20] B. Martinet. "Brève communication. Régularisation d'inéquations variationnelles par approximations successives". In: *Revue française d'informatique et de recherche opérationnelle. Série rouge* 4.R3 (1970), pp. 154–158.

[21] P. Tseng. "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *J. Optim. Theory Appl.* 109.3 (2001), pp. 475–494.

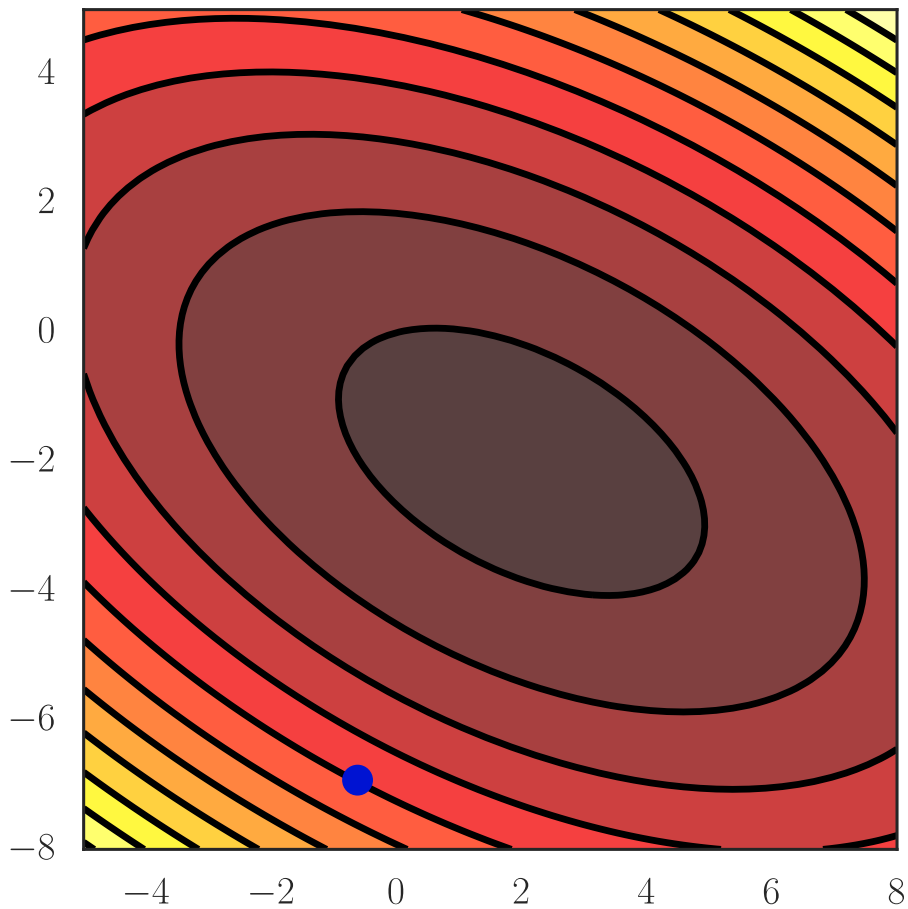
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



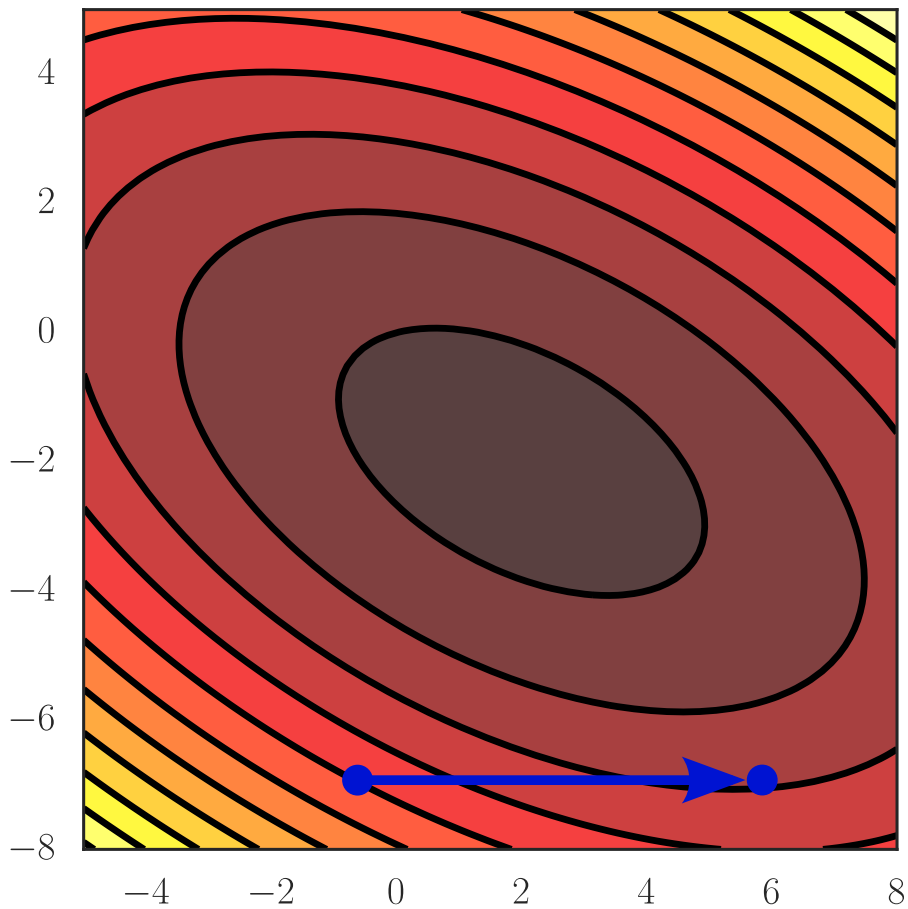
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



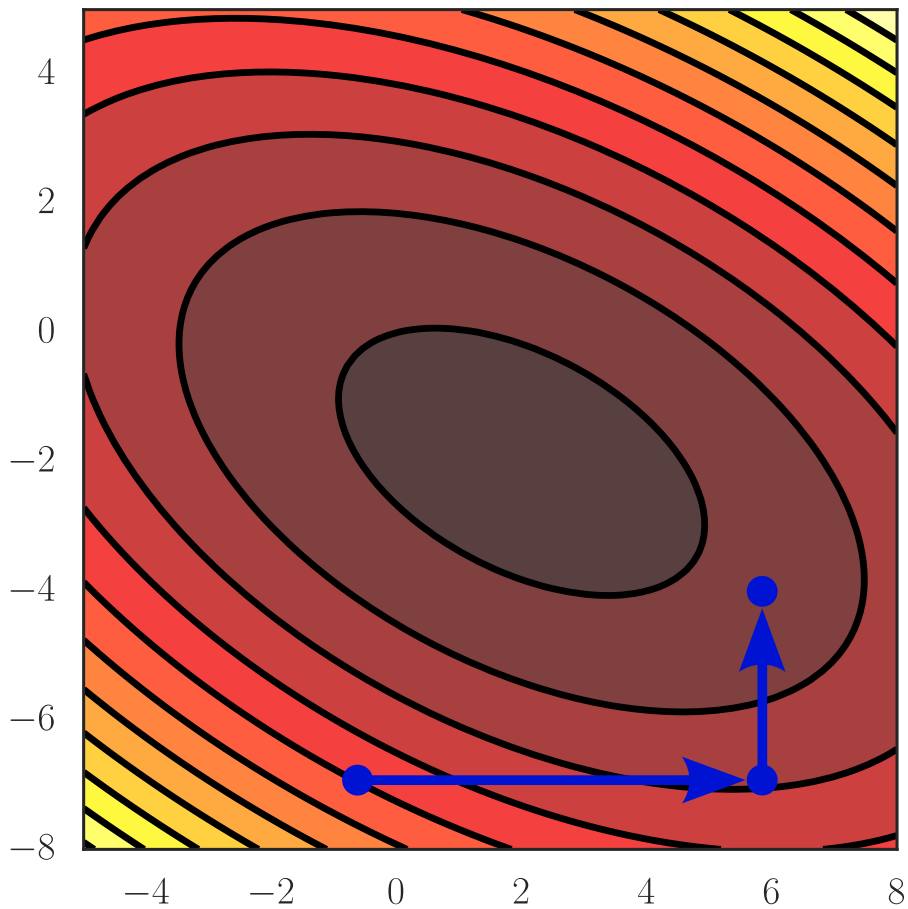
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



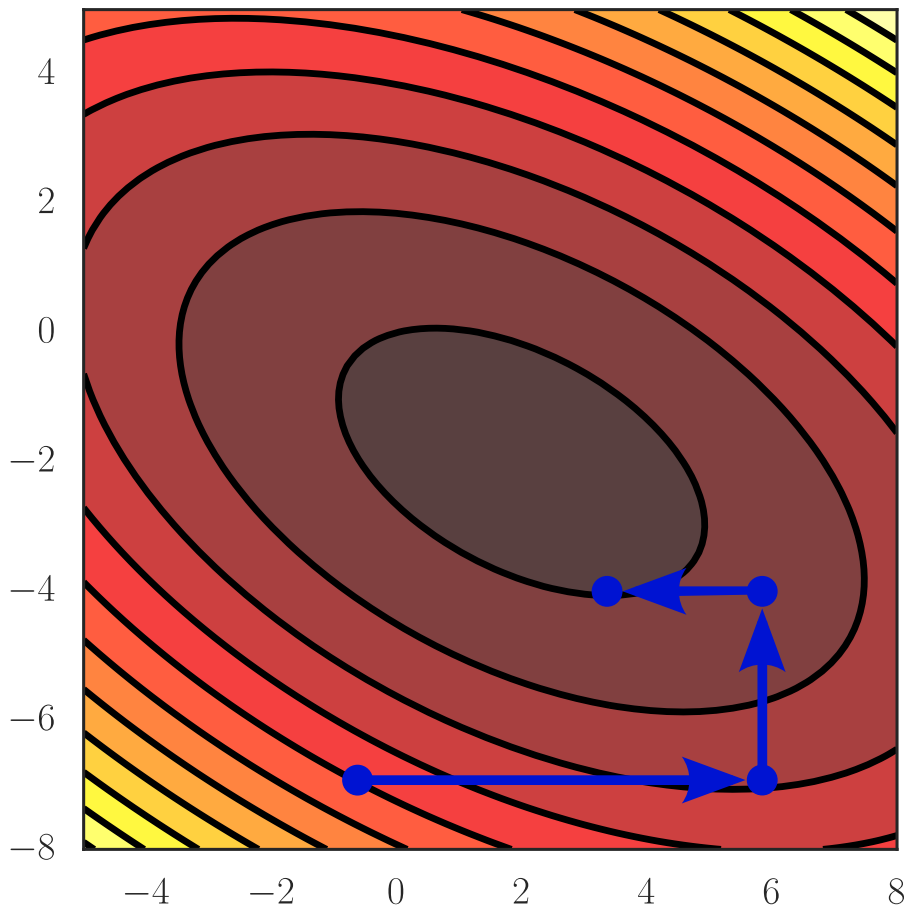
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



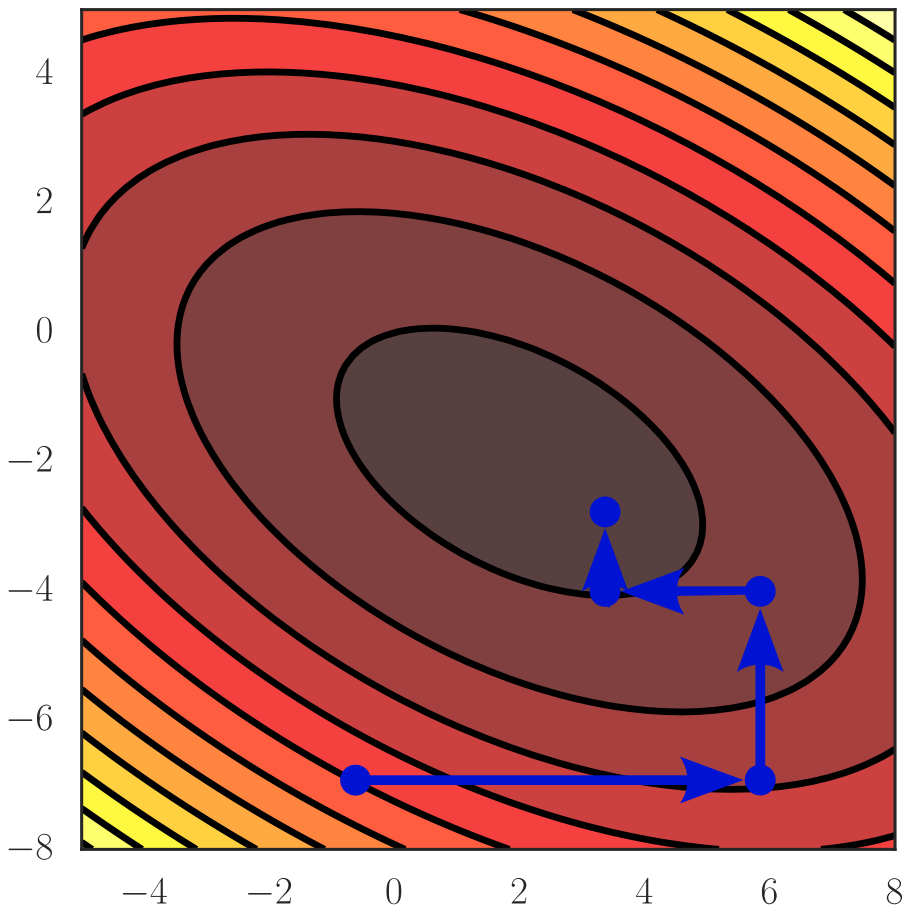
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



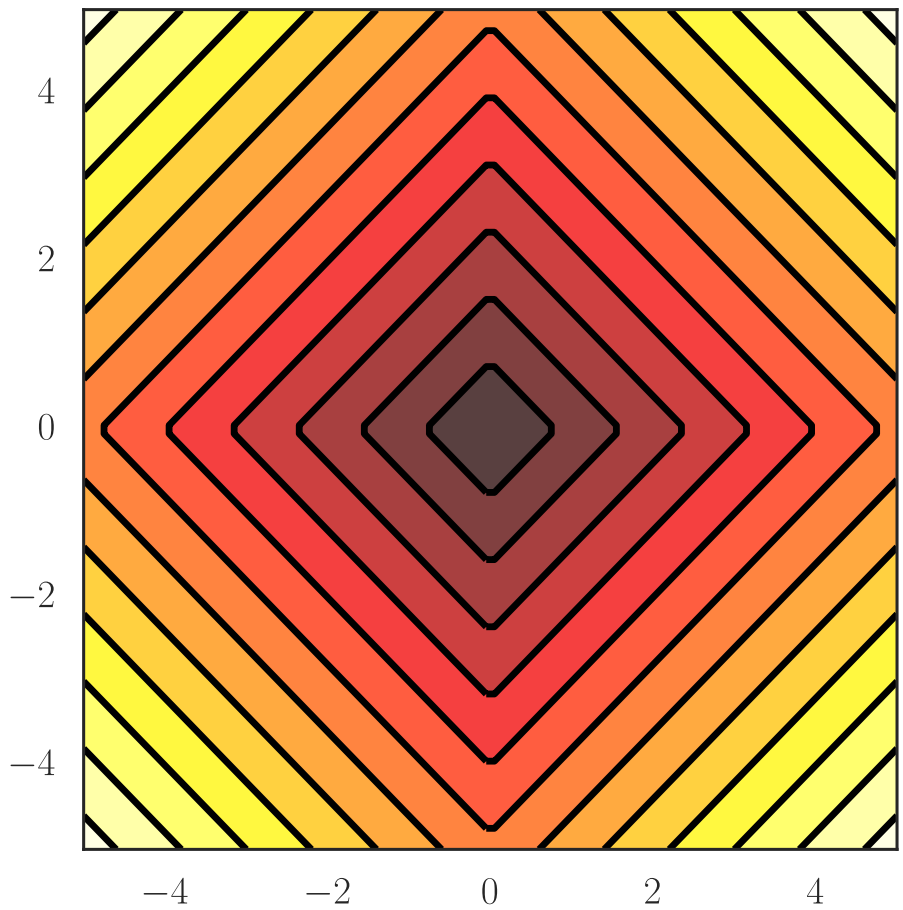
Examples (Convex case)

Convergence toward minimum for smooth case Tseng (2001)



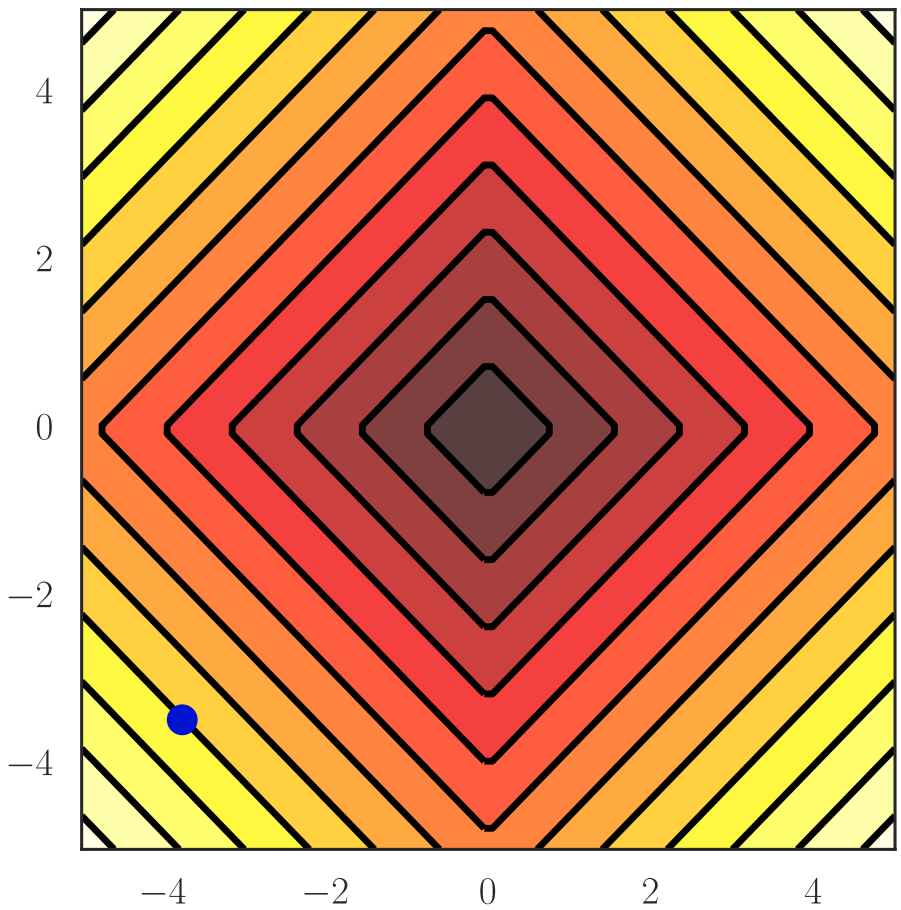
Examples (Convex case)

Convergence toward minimum for separable case Tseng (2001)



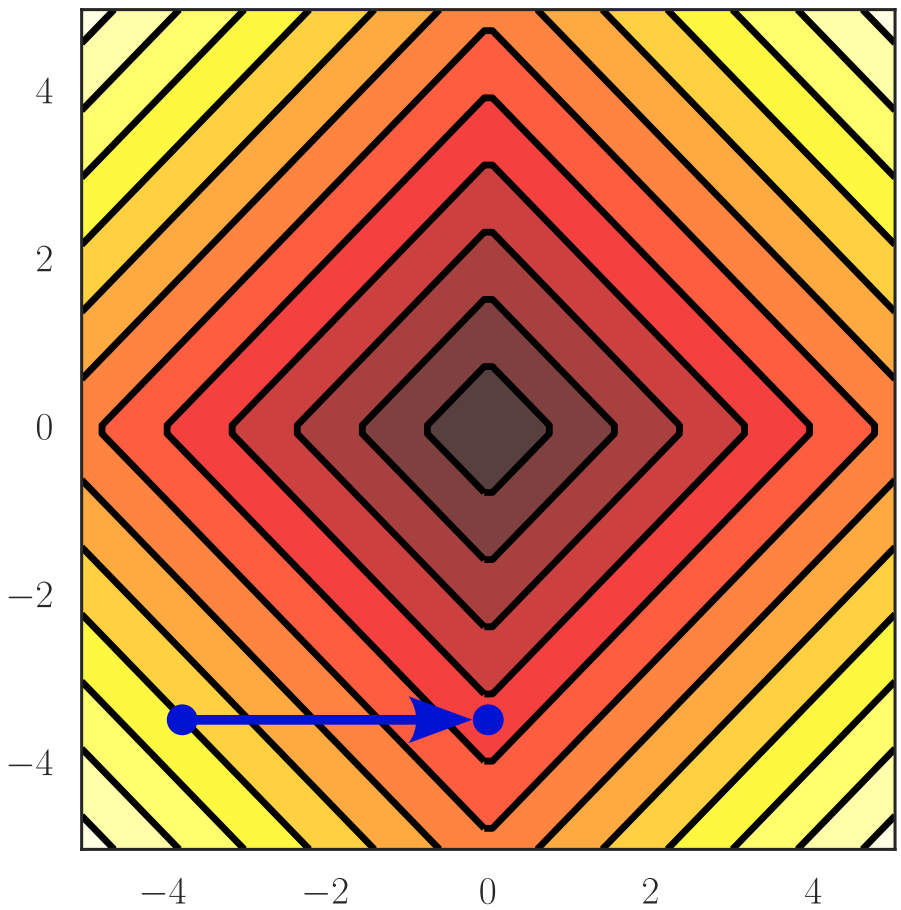
Examples (Convex case)

Convergence toward minimum for separable case Tseng (2001)



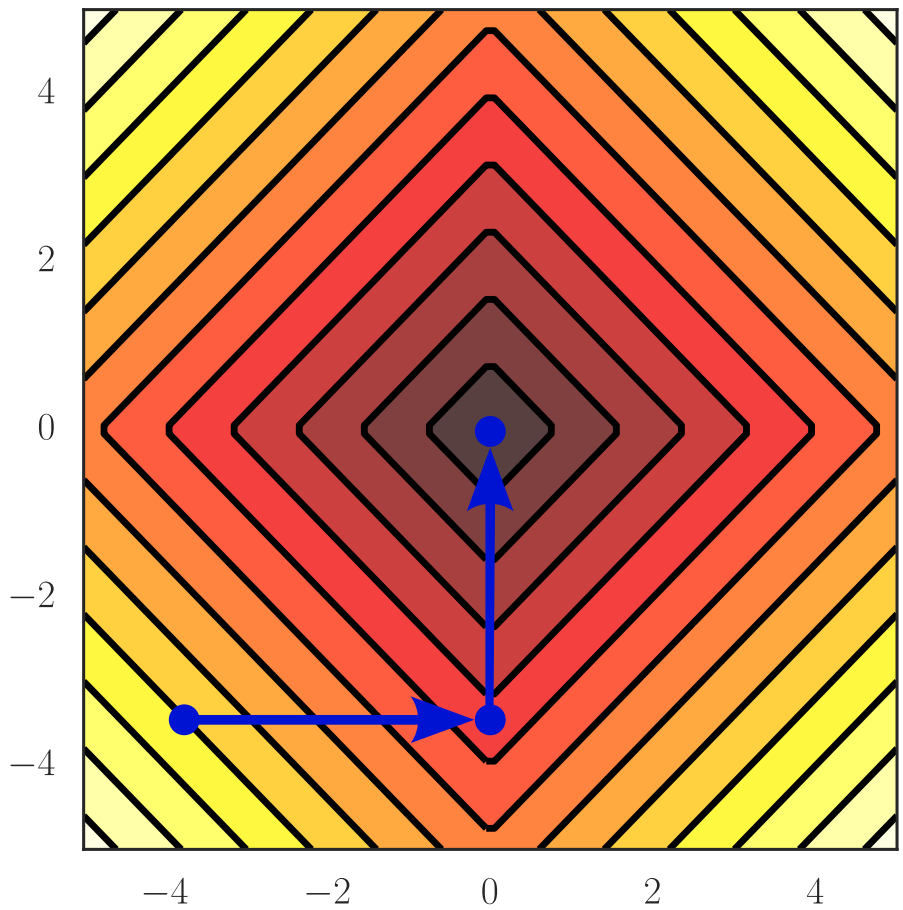
Examples (Convex case)

Convergence toward minimum for separable case Tseng (2001)



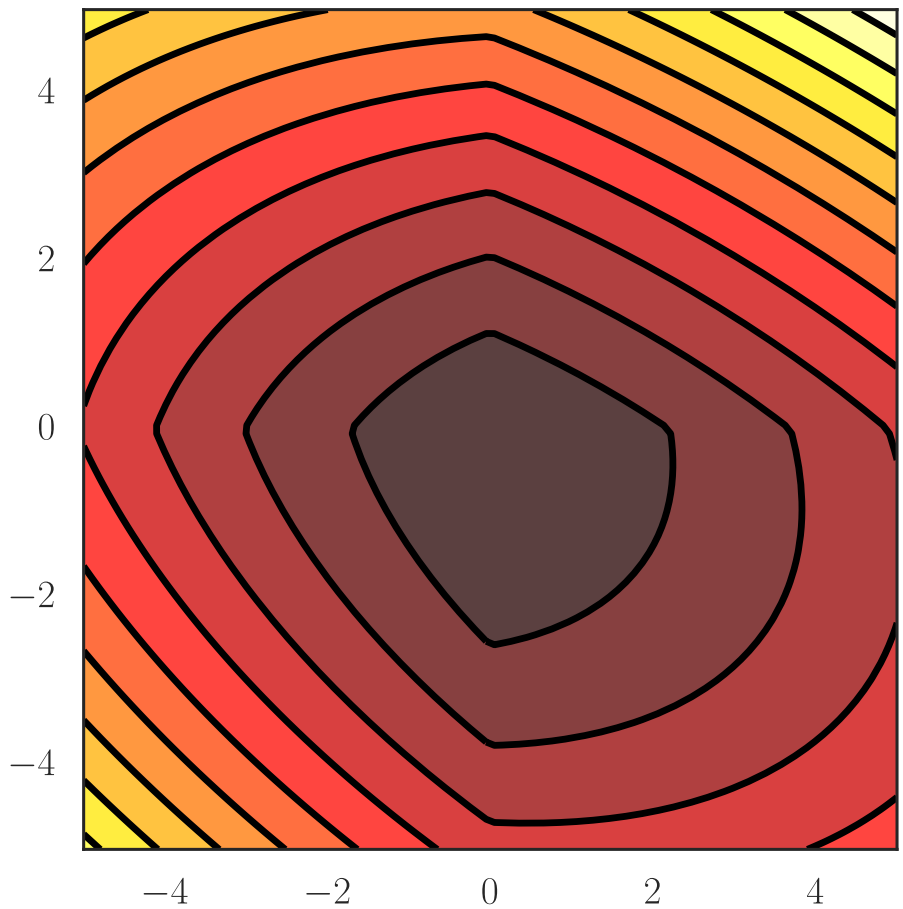
Examples (Convex case)

Convergence toward minimum for separable case Tseng (2001)



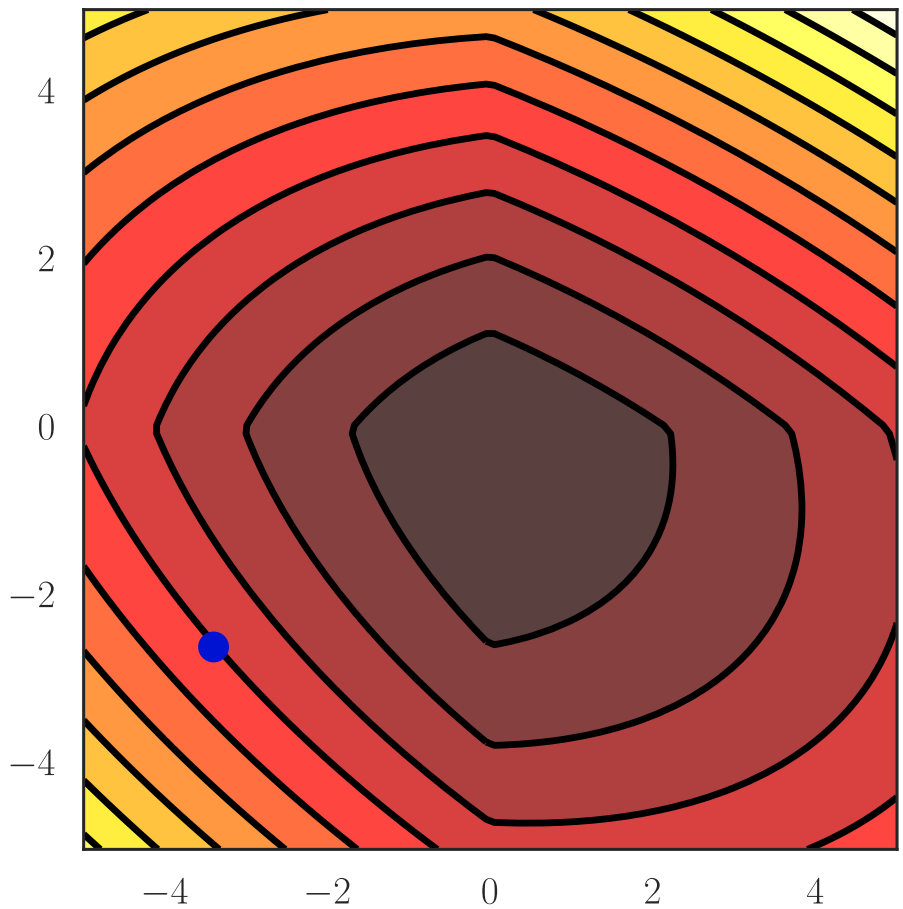
Examples (Convex case)

Convergence toward minimum smooth + separable case Tseng (2001)



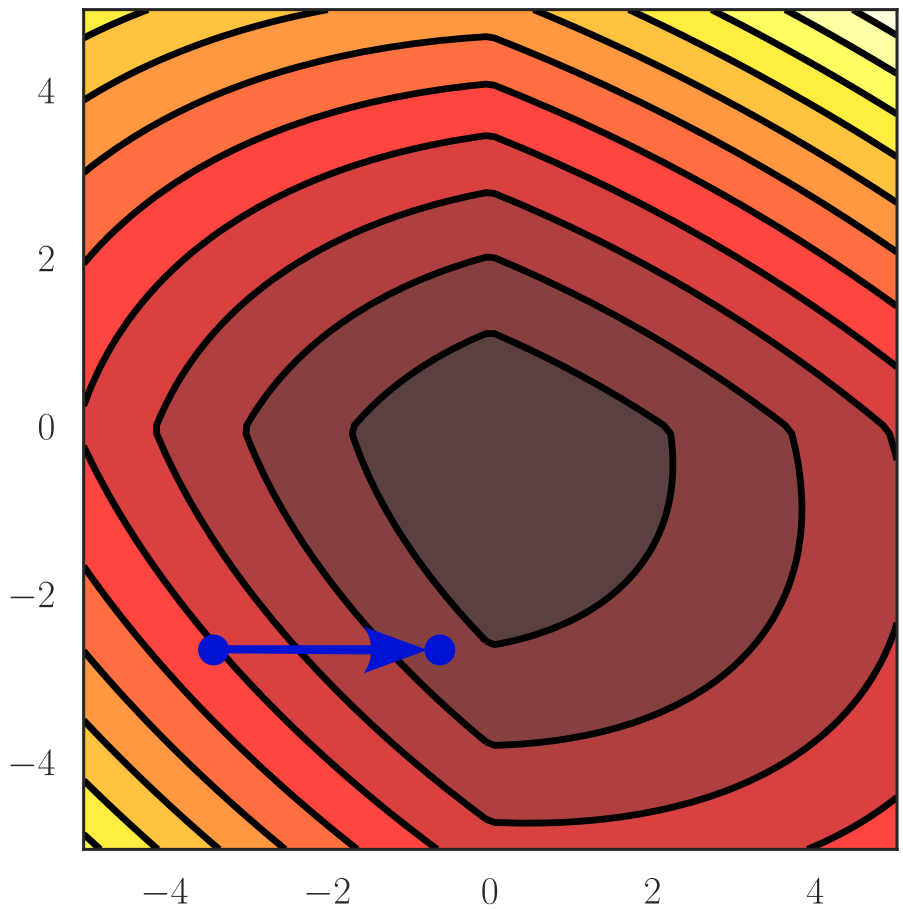
Examples (Convex case)

Convergence toward minimum smooth + separable case Tseng (2001)



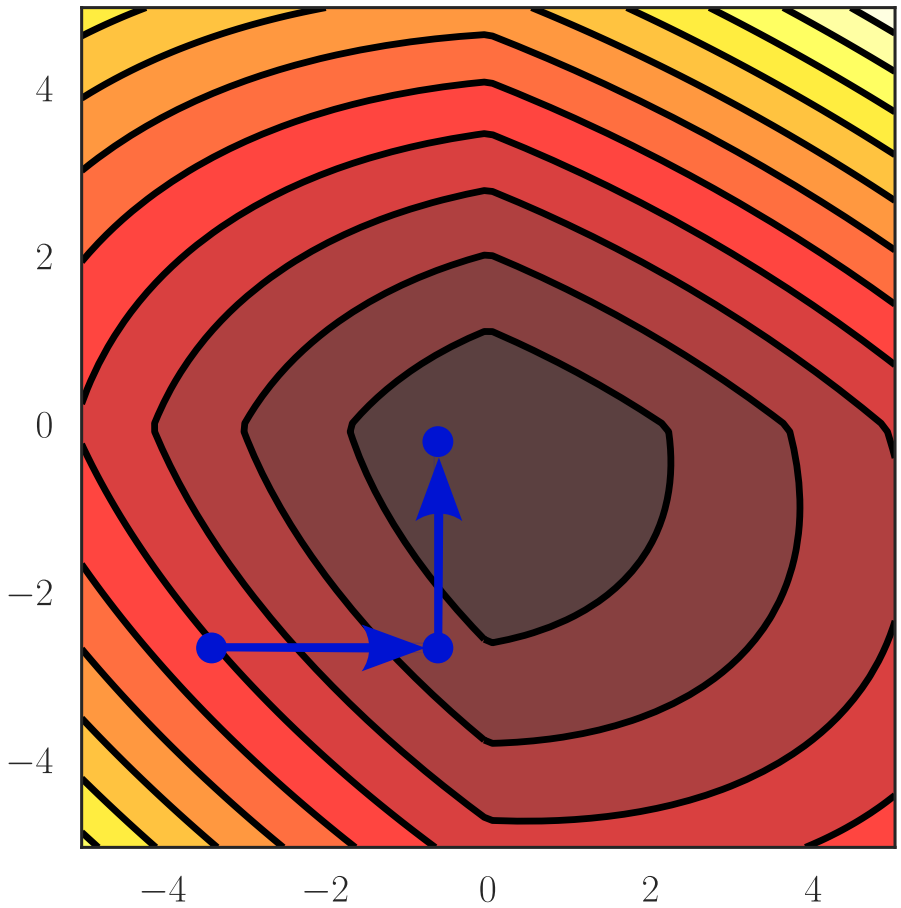
Examples (Convex case)

Convergence toward minimum smooth + separable case Tseng (2001)



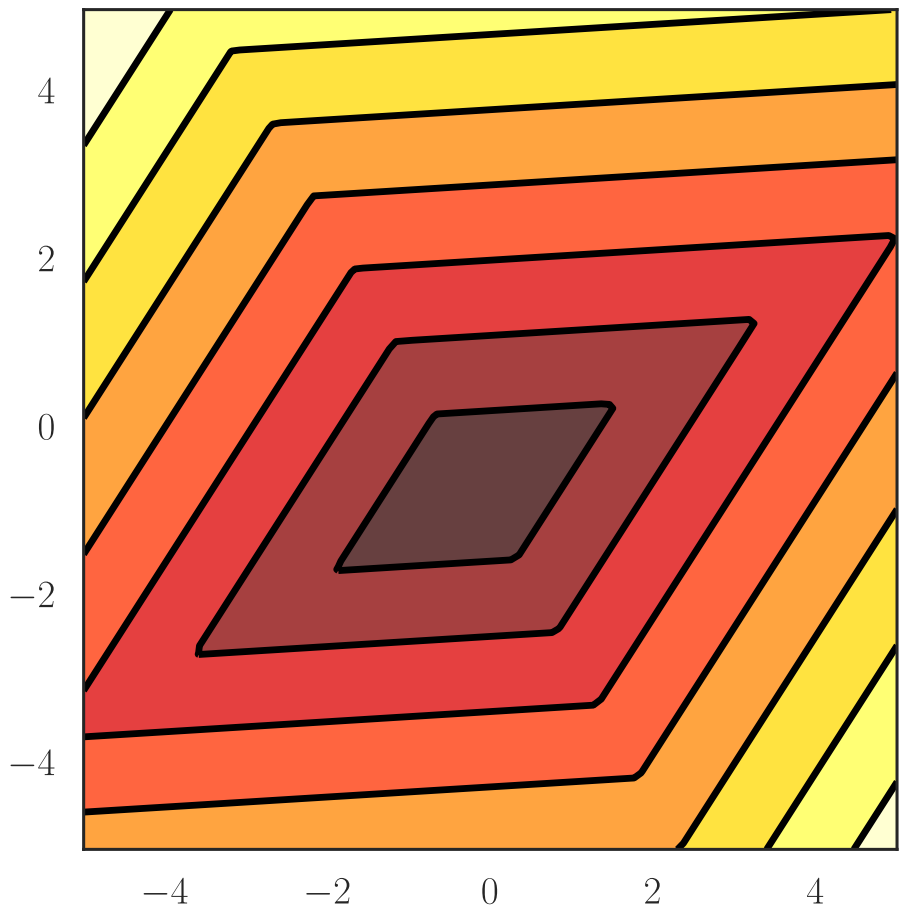
Examples (Convex case)

Convergence toward minimum smooth + separable case Tseng (2001)



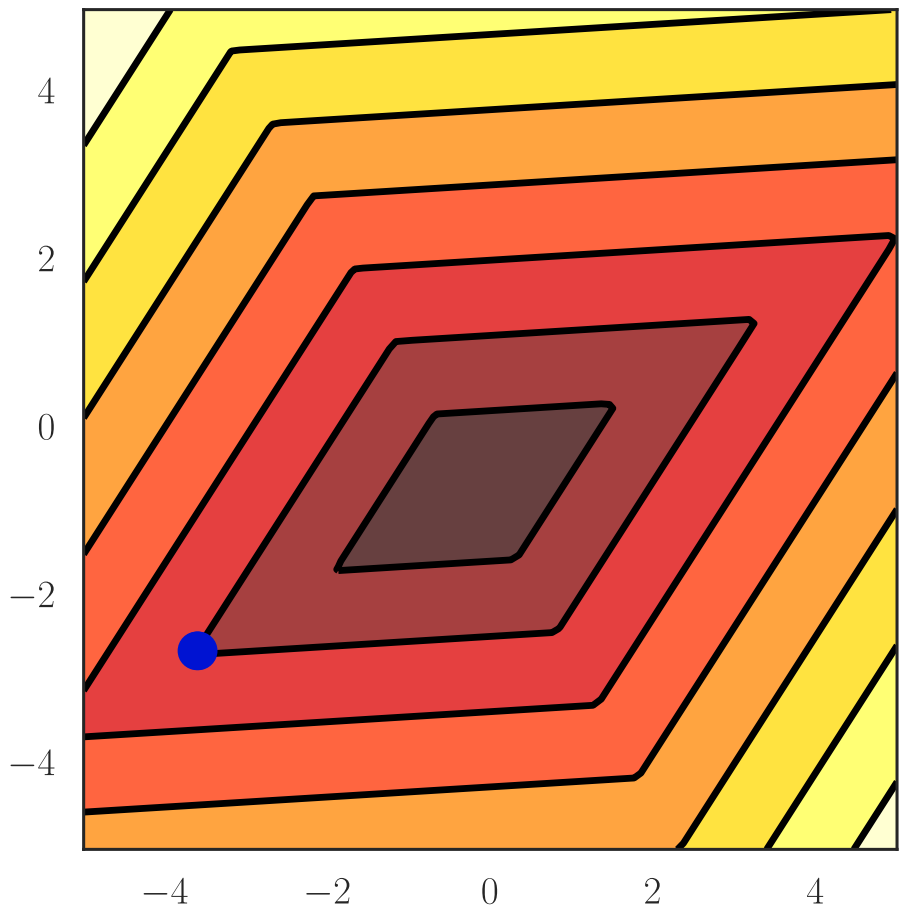
Examples (Convex case)

Beware: can fail on non-smooth / non separable cases



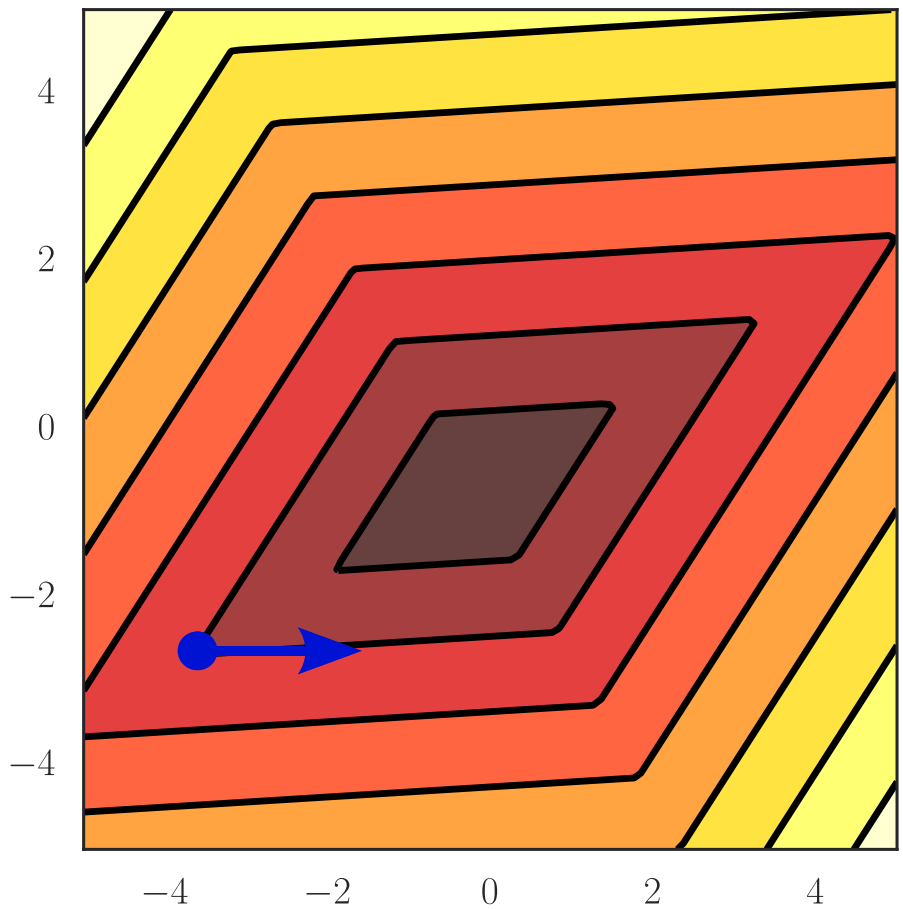
Examples (Convex case)

Beware: can fail on non-smooth / non separable cases



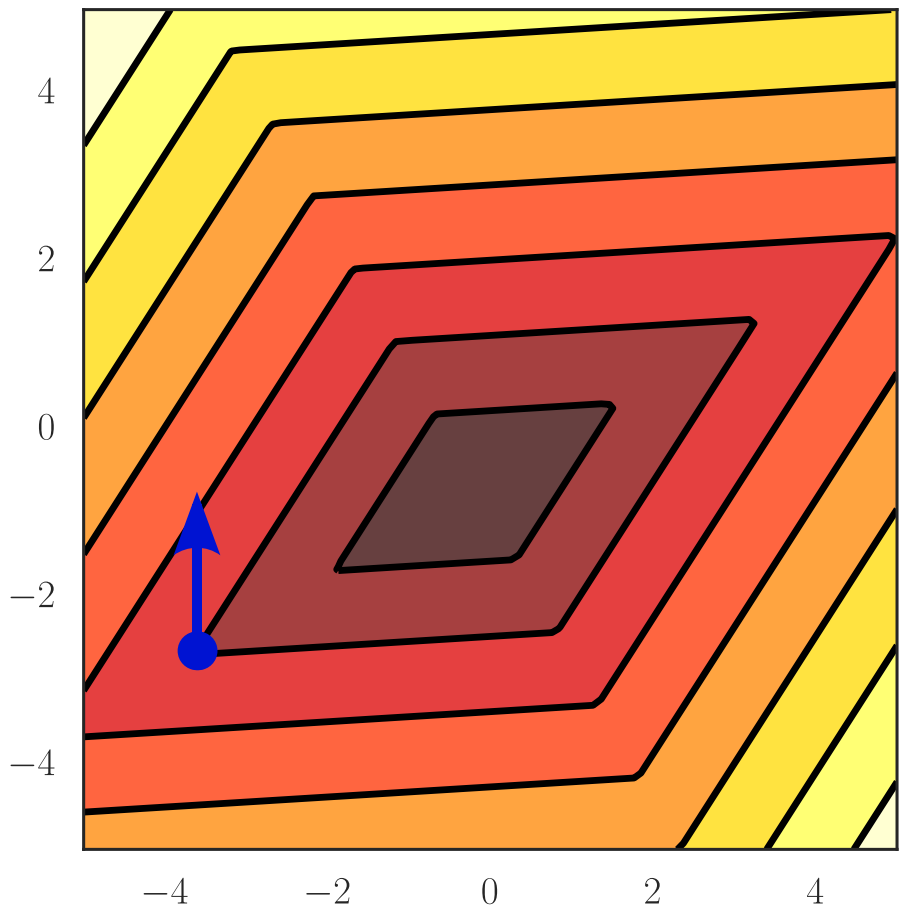
Examples (Convex case)

Beware: can fail on non-smooth / non separable cases



Examples (Convex case)

Beware: can fail on non-smooth / non separable cases



CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

for any $j \in [\![1, p]\!]$, do:

CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

$$\mathbf{r}^{\text{int}} \leftarrow \mathbf{r} + \mathbf{x}_j \beta_j$$

for any $j \in [\![1, p]\!]$, do:

CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

$$\mathbf{r}^{\text{int}} \leftarrow \mathbf{r} + \mathbf{x}_j \beta_j$$

for any $j \in [\![1, p]\!]$, do: $\quad \beta_j \leftarrow \mathrm{ST}_{\frac{n\lambda}{\|\mathbf{x}_j\|^2}} \left(\dfrac{\mathbf{x}_j^\top \mathbf{r}^{\text{int}}}{\|\mathbf{x}_j\|^2} \right)$

CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

$$\mathbf{r}^{\text{int}} \leftarrow \mathbf{r} + \mathbf{x}_j \beta_j$$

for any $j \in [\![1, p]\!]$, do: $\quad \beta_j \leftarrow \mathrm{ST}_{\frac{n\lambda}{\|\mathbf{x}_j\|^2}} \left(\dfrac{\mathbf{x}_j^\top \mathbf{r}^{\text{int}}}{\|\mathbf{x}_j\|^2} \right)$

$$r \leftarrow \mathbf{r}^{\text{int}} - \mathbf{x}_j \beta_j$$

CD for Lasso

Lazy update: maintain **residual** $\mathbf{r} = \mathbf{y} - X\boldsymbol{\beta}$ and coeff. $\boldsymbol{\beta}$

$$\mathbf{r}^{\text{int}} \leftarrow \mathbf{r} + \mathbf{x}_j \beta_j$$

for any $j \in [\![1, p]\!]$, do: $\quad \beta_j \leftarrow \mathrm{ST}_{\frac{n\lambda}{\|\mathbf{x}_j\|^2}} \left(\dfrac{\mathbf{x}_j^\top \mathbf{r}^{\text{int}}}{\|\mathbf{x}_j\|^2} \right)$

$$r \leftarrow \mathbf{r}^{\text{int}} - \mathbf{x}_j \beta_j$$

Low memory footprint:

- store residual vector: size n
- store coeff. vector: size p

Rem: generally normalized features $\|\mathbf{x}_j\|_2^2 = 1$ or $\|\mathbf{x}_j\|_2^2 = n$

Outline

Speeding-up solvers

Two approaches:

- safe screening[22],[23] (**backward approach**): remove feature j when it is certified that $\hat{\beta}_j = 0$
- working set[24] (**forward approach**): focus on j's for which it is very likely that $\hat{\beta}_j \neq 0$.

We recommend the later: `celer`
https://github.com/mathurinm/celer

---

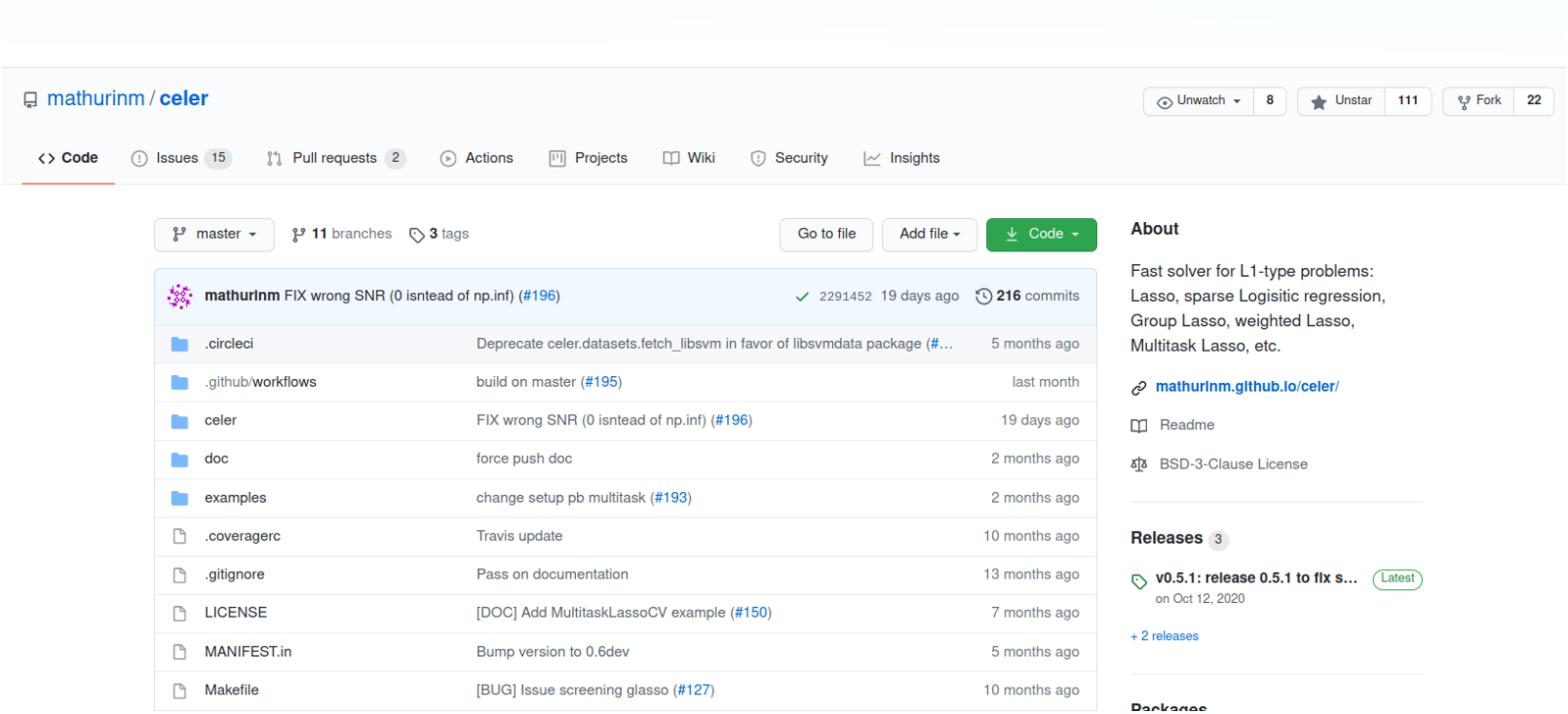
[22] L. El Ghaoui, V. Viallon, and T. Rabbani. "Safe feature elimination in sparse supervised learning". In: *J. Pacific Optim.* 8.4 (2012), pp. 667–698.

[23] A. Bonnefoy et al. "A dynamic screening principle for the lasso". In: *EUSIPCO*. 2014, pp. 6–10.

[24] T. B. Johnson and C. Guestrin. "Blitz: A Principled Meta-Algorithm for Scaling Sparse Optimization". In: *ICML*. vol. 37. 2015, pp. 1171–1179.

Reusable science

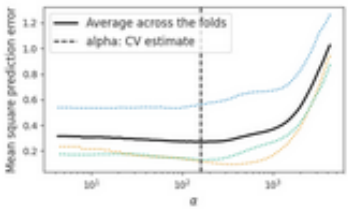
https://github.com/mathurinm/celer:
code with continuous integration, code coverage, bug tracker



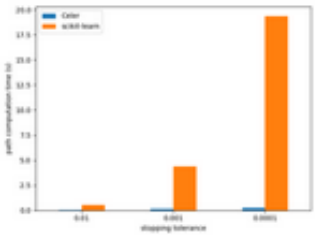
Examples gallery:

https://mathurinm.github.io/celer: documentation (examples, API)

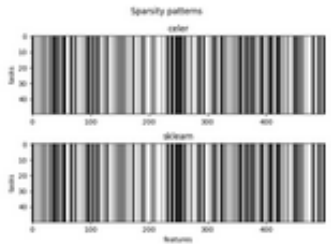
Use LogisticRegression class with Celer and Prox-Newton solvers



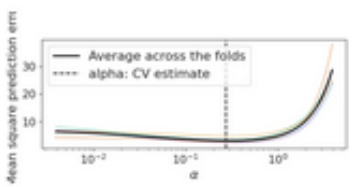
Run LassoCV for cross-validation on the Leukemia dataset



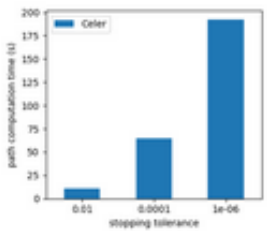
Lasso path computation on Leukemia dataset



Run MultitaskLassoCV and compare performance with scikit-learn



Run GroupLasso and GroupLasso CV for structured sparse recovery



Lasso path computation on Finance/log1p dataset

Drop-in sklearn replacement

```
from sklearn.linear_model import Lasso, LassoCV
from celer import Lasso, LassoCV
```

celer.Lasso

class `celer.` **Lasso** (***alpha=1.0, max_iter=100, gap_freq=10, max_epochs=50000, p0=10, verbose=0, tol=1e-06, prune=0, fit_intercept=True***)

Lasso scikit-learn estimator based on Celer solver

The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - X beta||^2_2 + alpha * ||beta||_1
```

Parameters:

alpha : float, optional

> Constant that multiplies the L1 term. Defaults to 1.0. `alpha = 0` is equivalent to an ordinary least square. For numerical reasons, using `alpha = 0` with the `Lasso` object is not advised.

max_iter : int, optional

> The maximum number of iterations (subproblem definitions)

gap_freq : int

> Number of coordinate descent epochs between each duality gap computations.

max_epochs : int

Drop-in sklearn replacement

```
from sklearn.linear_model import Lasso, LassoCV
from celer import Lasso, LassoCV
```

From 10,000 s to 50 s for cross-validation on Finance dataset

celer.Lasso

Fork me on GitHub

class `celer.` **Lasso** (***alpha=1.0, max_iter=100, gap_freq=10, max_epochs=50000, p0=10, verbose=0, tol=1e-06, prune=0, fit_intercept=True***)

Lasso scikit-learn estimator based on Celer solver

The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - X beta||^2_2 + alpha * ||beta||_1
```

Parameters:

alpha : float, optional

Constant that multiplies the L1 term. Defaults to 1.0. `alpha = 0` is equivalent to an ordinary least square. For numerical reasons, using `alpha = 0` with the `Lasso` object is not advised.

max_iter : int, optional

The maximum number of iterations (subproblem definitions)

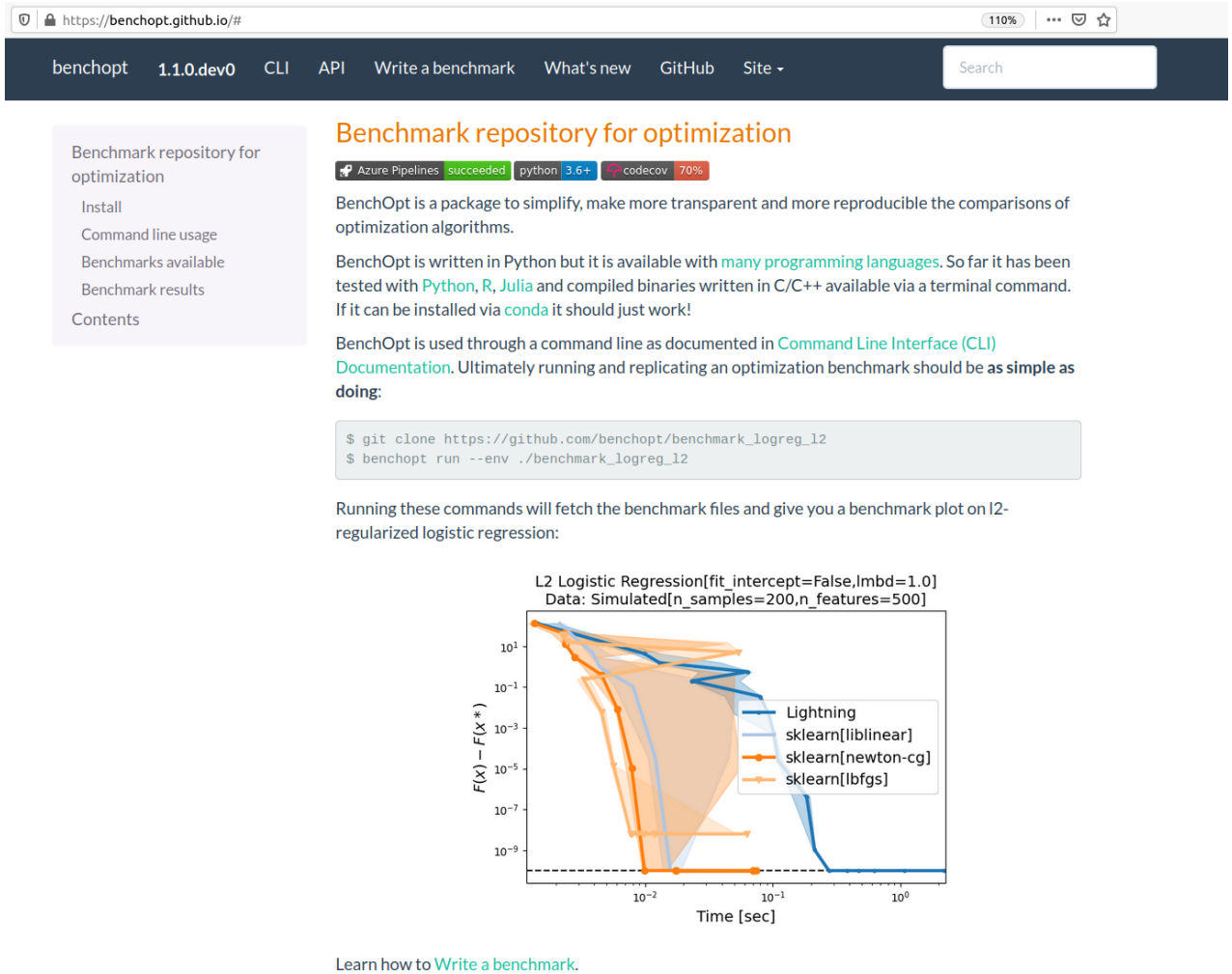
gap_freq : int

Number of coordinate descent epochs between each duality gap computations.

max_epochs : int

Try on your favorite dataset: `benchopt`

A fastest optimization solver is dataset dependent $\Longrightarrow$
benchmark solvers with https://benchopt.github.io/



See also: https://benchopt.github.io/results/

Outline

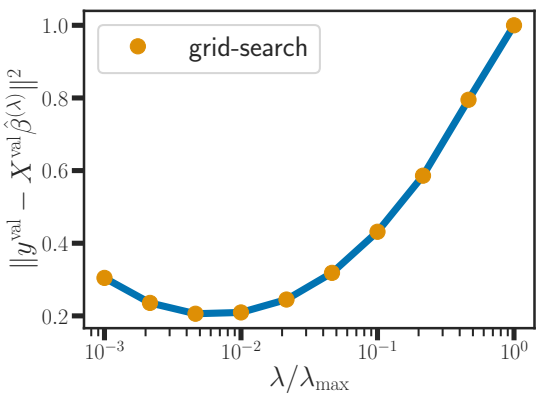
Hyperparameter optimization (HO)

We present here hold-out validation[(25)] $X, \mathbf{y}$ split into $X^{\text{train}}, \mathbf{y}^{\text{train}}$ and $X^{\text{test}}, \mathbf{y}^{\text{test}}$

[(25)] cross-validation works similarly, but less notationally friendly

Hyperparameter optimization (HO)

We present here hold-out validation[25] $X, \mathbf{y}$ split into $X^{\text{train}}, \mathbf{y}^{\text{train}}$ and $X^{\text{test}}, \mathbf{y}^{\text{test}}$



Real-sim dataset
Validation loss w.r.t. λ

Lasso example:

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2}{2n} + \lambda\|\beta\|_1$$

Criterion: held-out loss

$$\arg\min_{\lambda \in \mathbb{R}} \|\mathbf{y}^{\text{test}} - X^{\text{val}}\hat{\beta}^{(\lambda)}\|^2$$

[25] cross-validation works similarly, but less notationally friendly

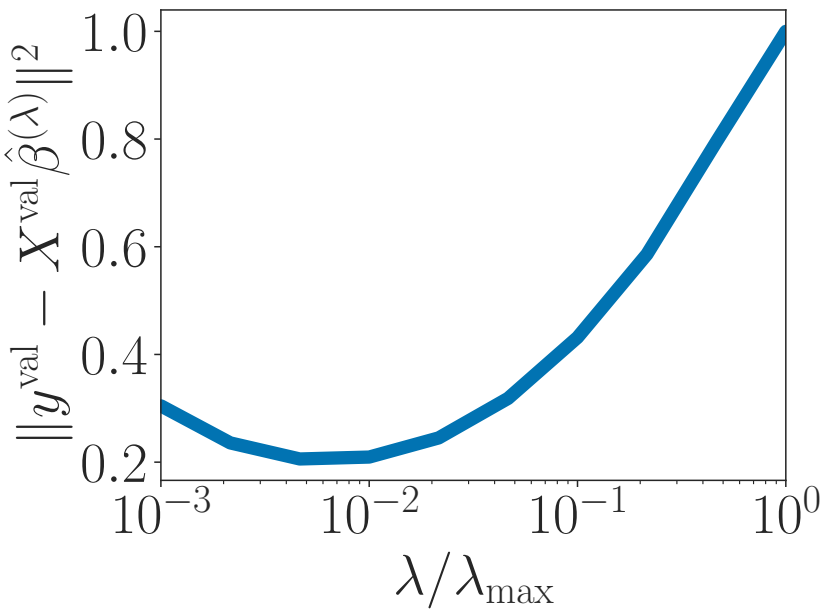
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

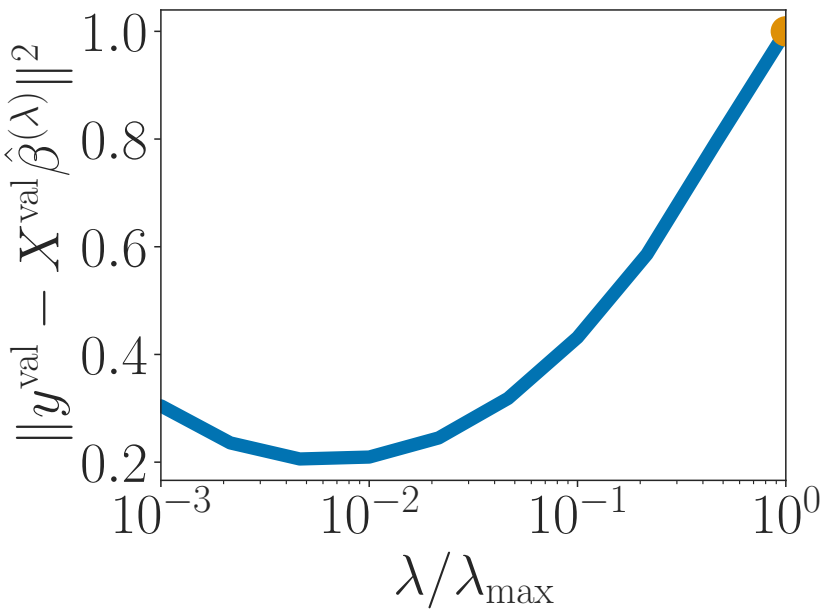
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

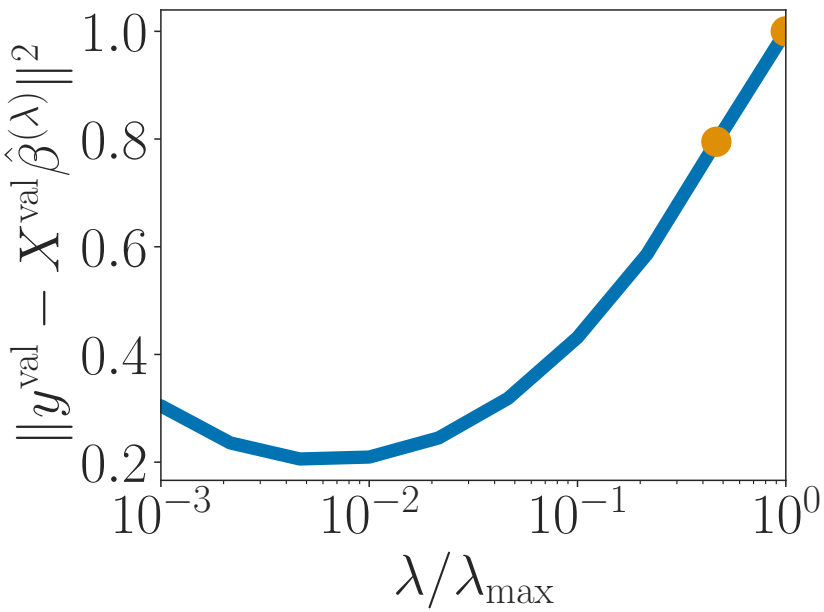
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

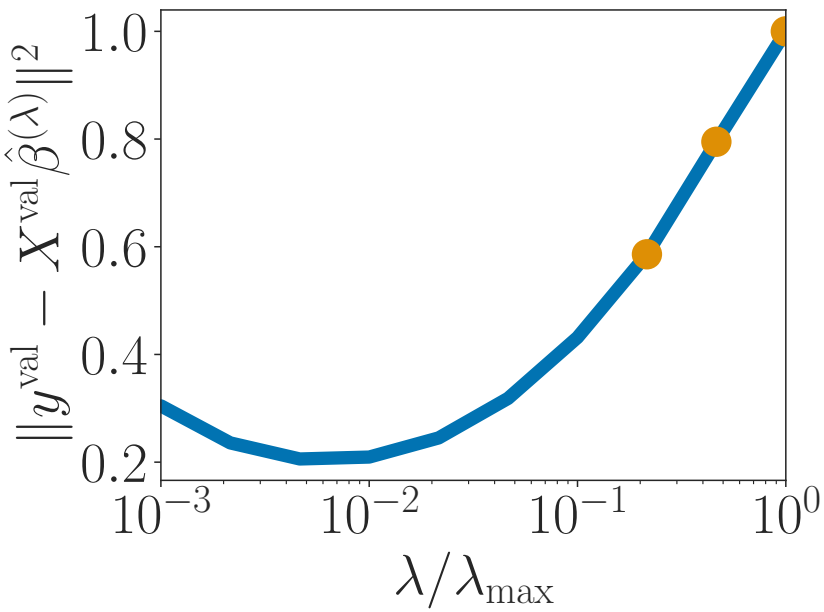
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

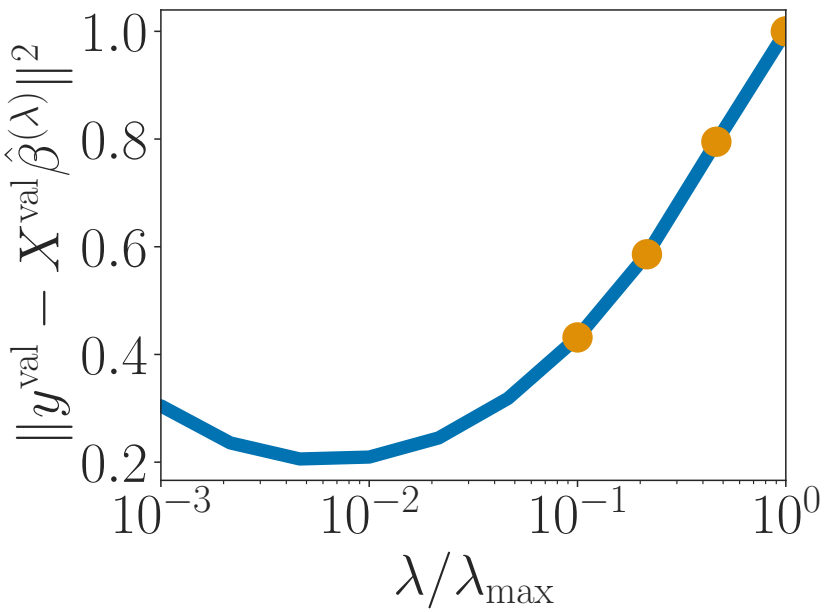
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

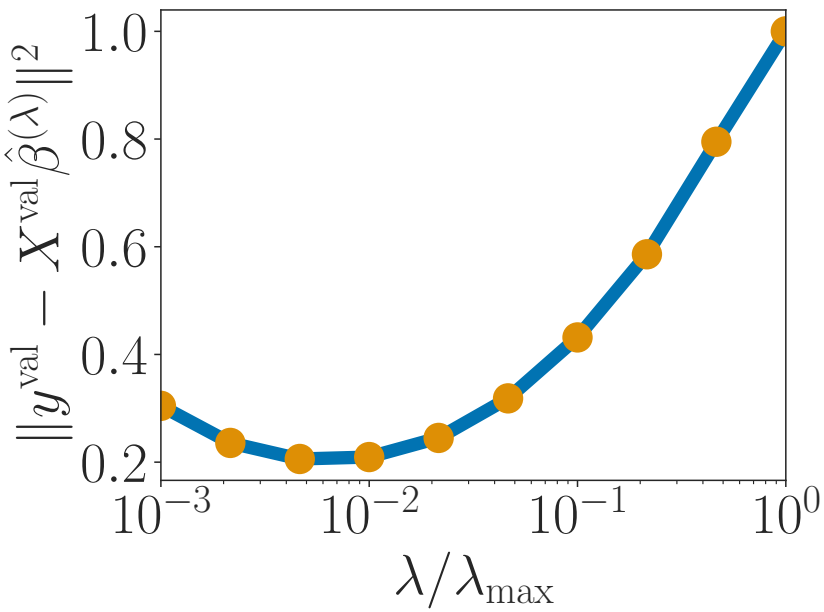
HO as a bilevel optimization problem[26][27]

outer optimization problem

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

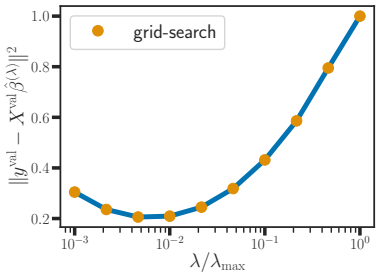
inner optimization problem



[26] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[27] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

Grid-search as a 0-order optimization method



$$\underset{\lambda \in \mathbb{R}}{\arg\min} \left\{ \mathcal{L}(\lambda) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}(\lambda)\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}(\lambda) \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

- Grid-search, random-search,[28] SMBO[29]:
  0-order methods to solve bilevel optimization problem
- **Idea:** if $\mathcal{L}$ is differentiable, use first order optimization,
  i.e., compute $\nabla_\lambda \mathcal{L}$
- Once $\nabla_\lambda \mathcal{L}(\lambda)$ is computed, use gradient descent[30]:
  $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda \mathcal{L}(\lambda^{(t)})$ with suitable $\rho > 0$

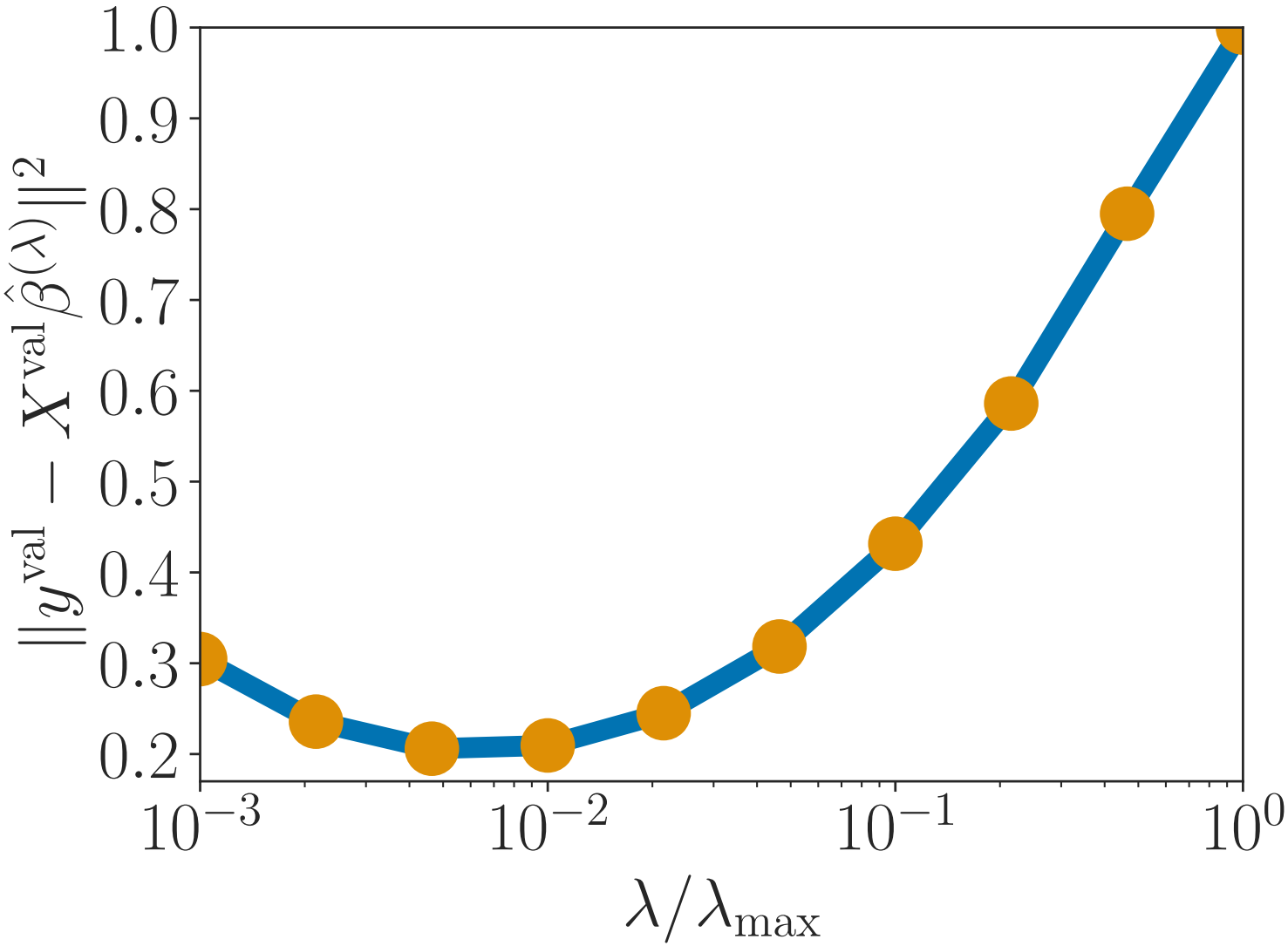
---

[28] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". In: *J. Mach. Learn. Res.* 13.2 (2012).

[29] E. Brochu, V. M. Cora, and N. De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: (2010).

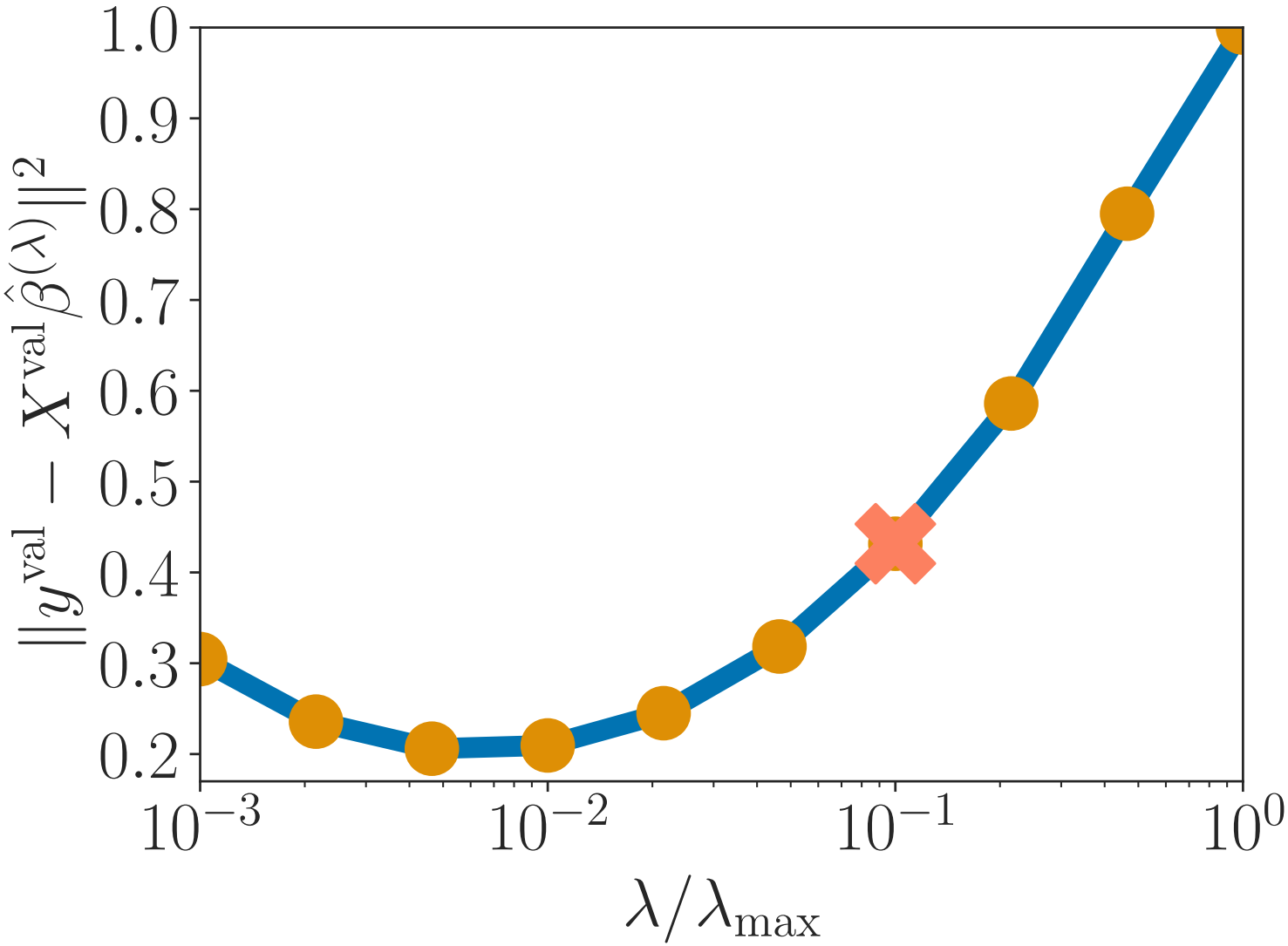
[30] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: *ICML*. vol. 48. 2016, pp. 737–746.

First order optimization in λ, Lasso



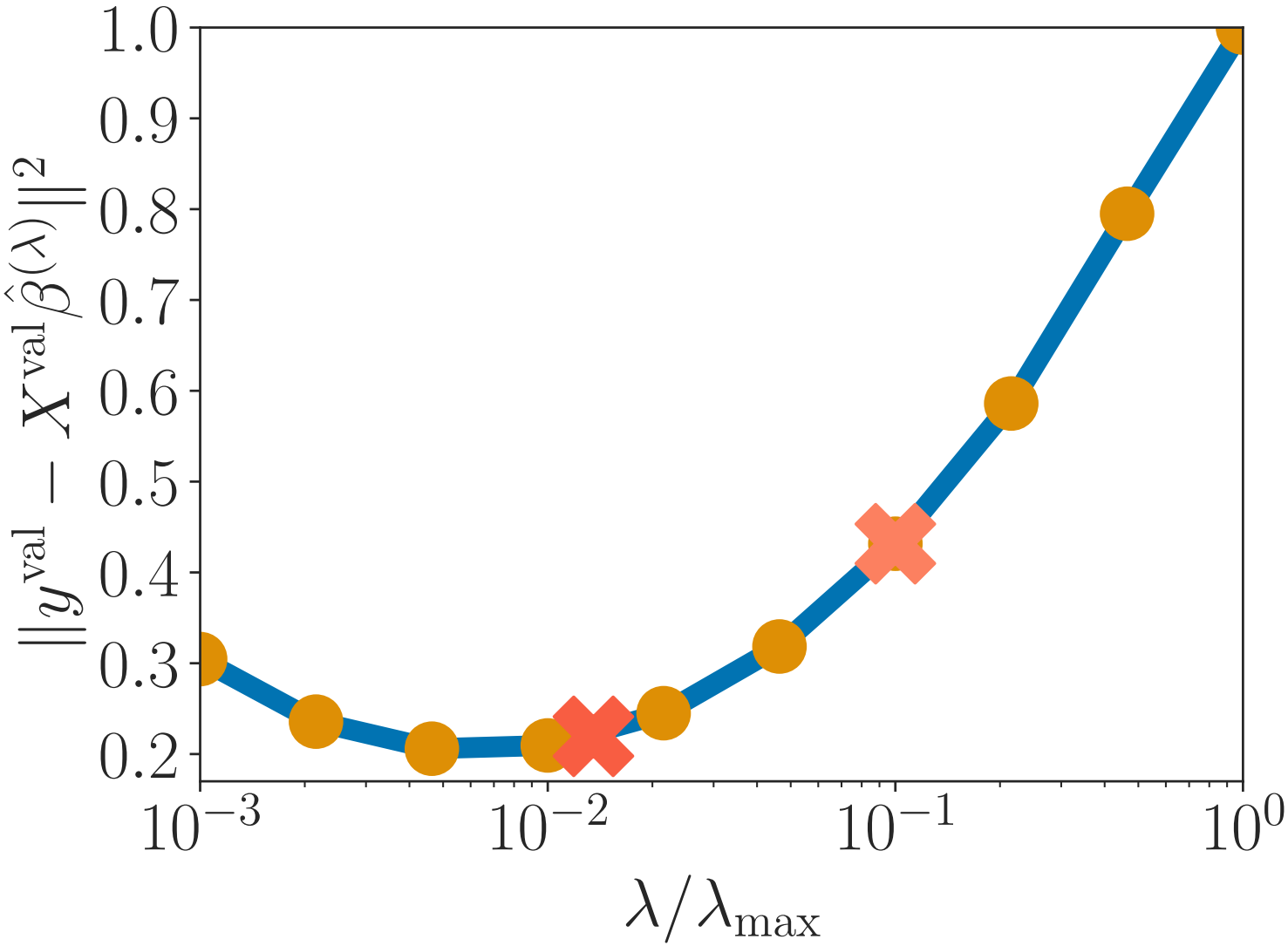
Real-sim dataset. Validation loss as a function of λ.

First order optimization in λ, Lasso



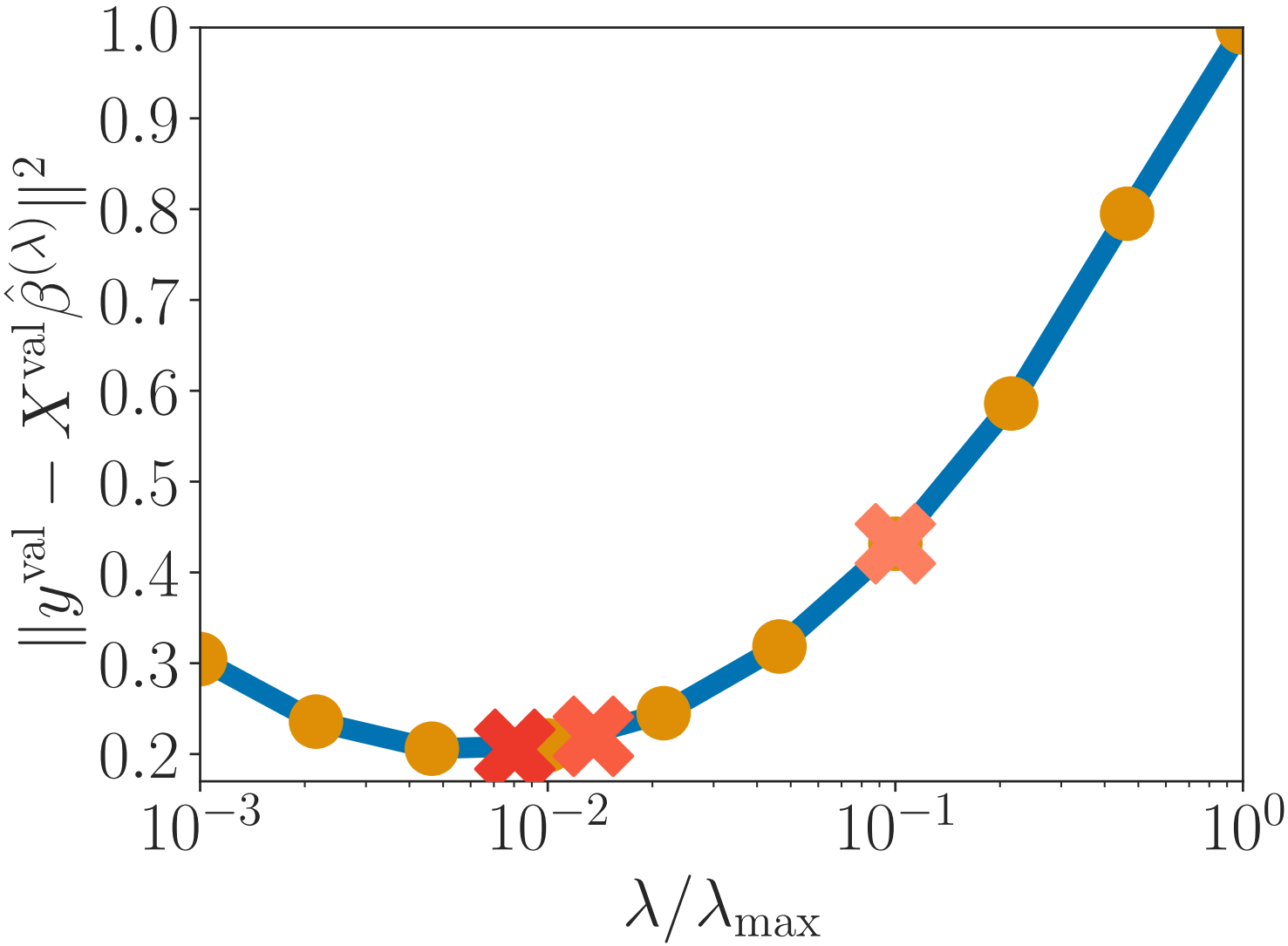
Real-sim dataset. Validation loss as a function of λ.

First order optimization in λ, Lasso



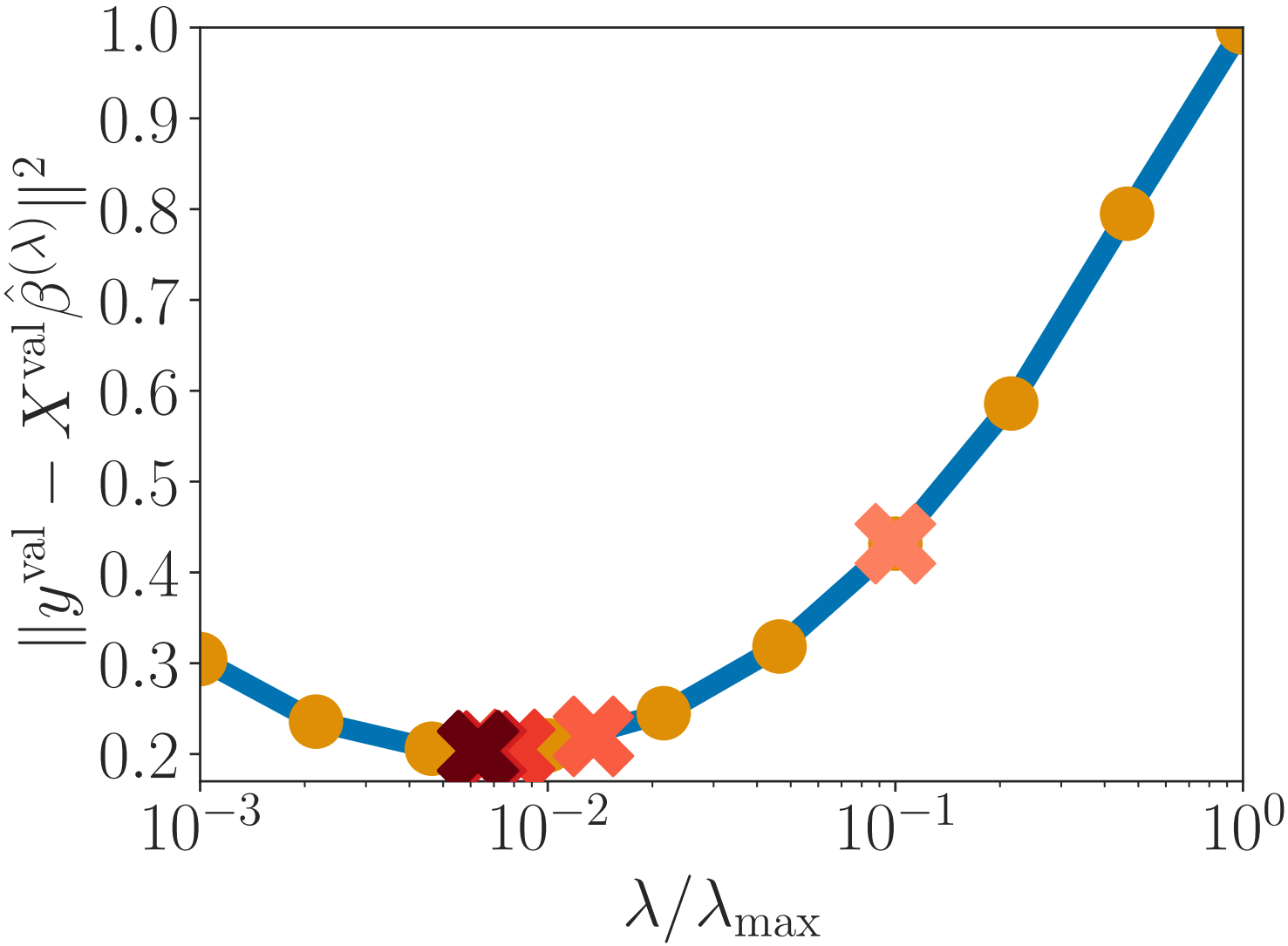
Real-sim dataset. Validation loss as a function of λ.

First order optimization in λ, Lasso



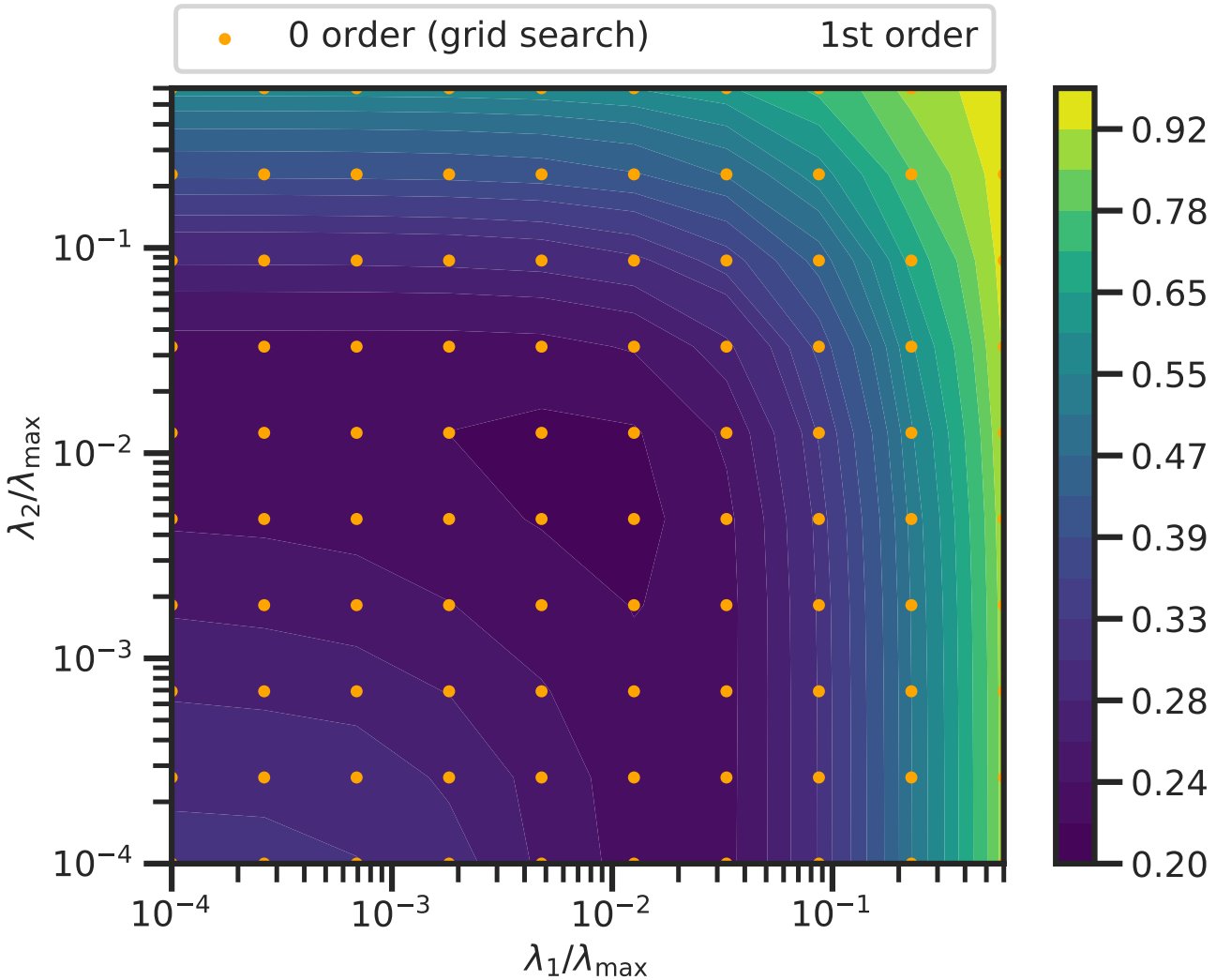
Real-sim dataset. Validation loss as a function of λ.

First order optimization in λ, Lasso



Real-sim dataset. Validation loss as a function of λ.

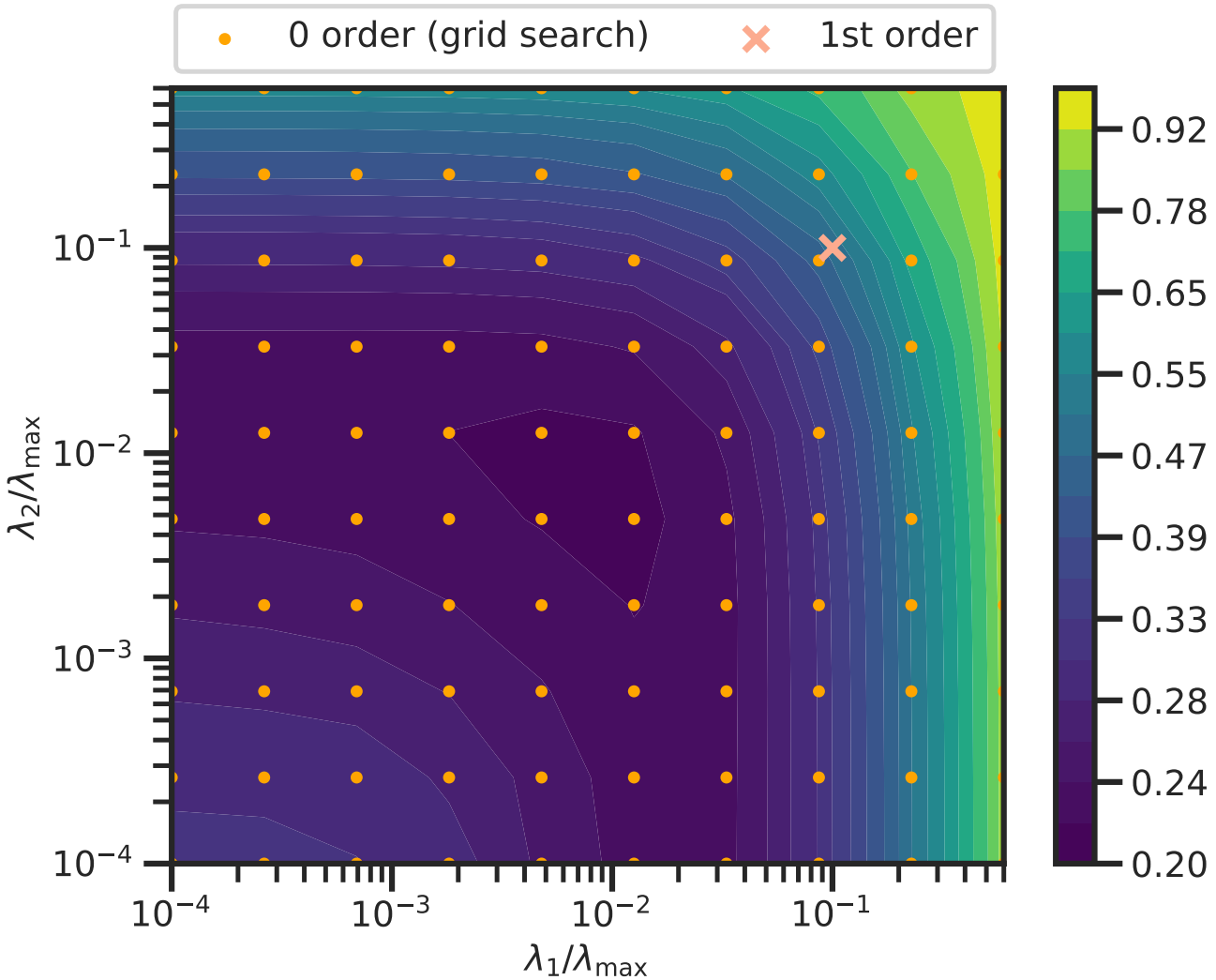
First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

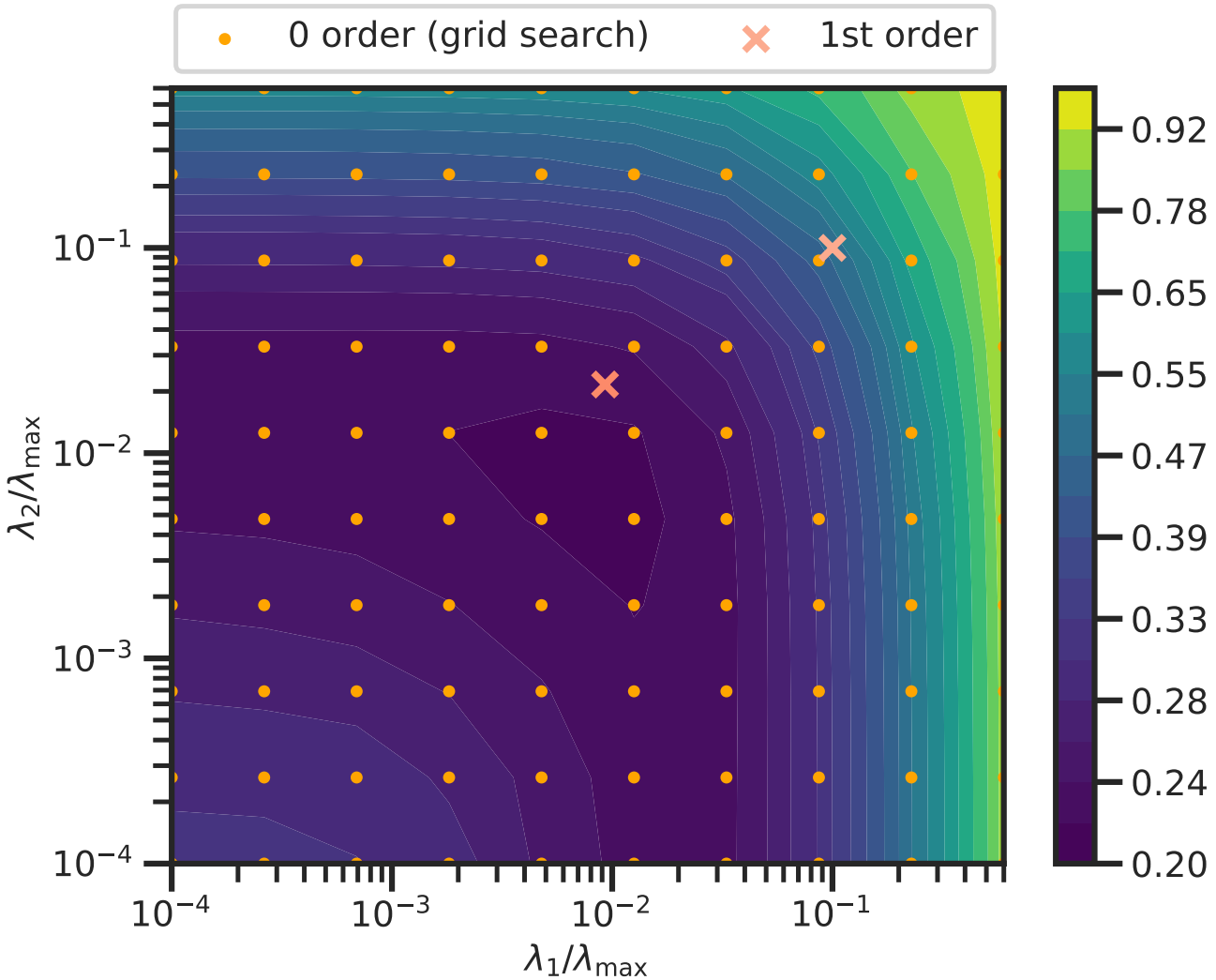
First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

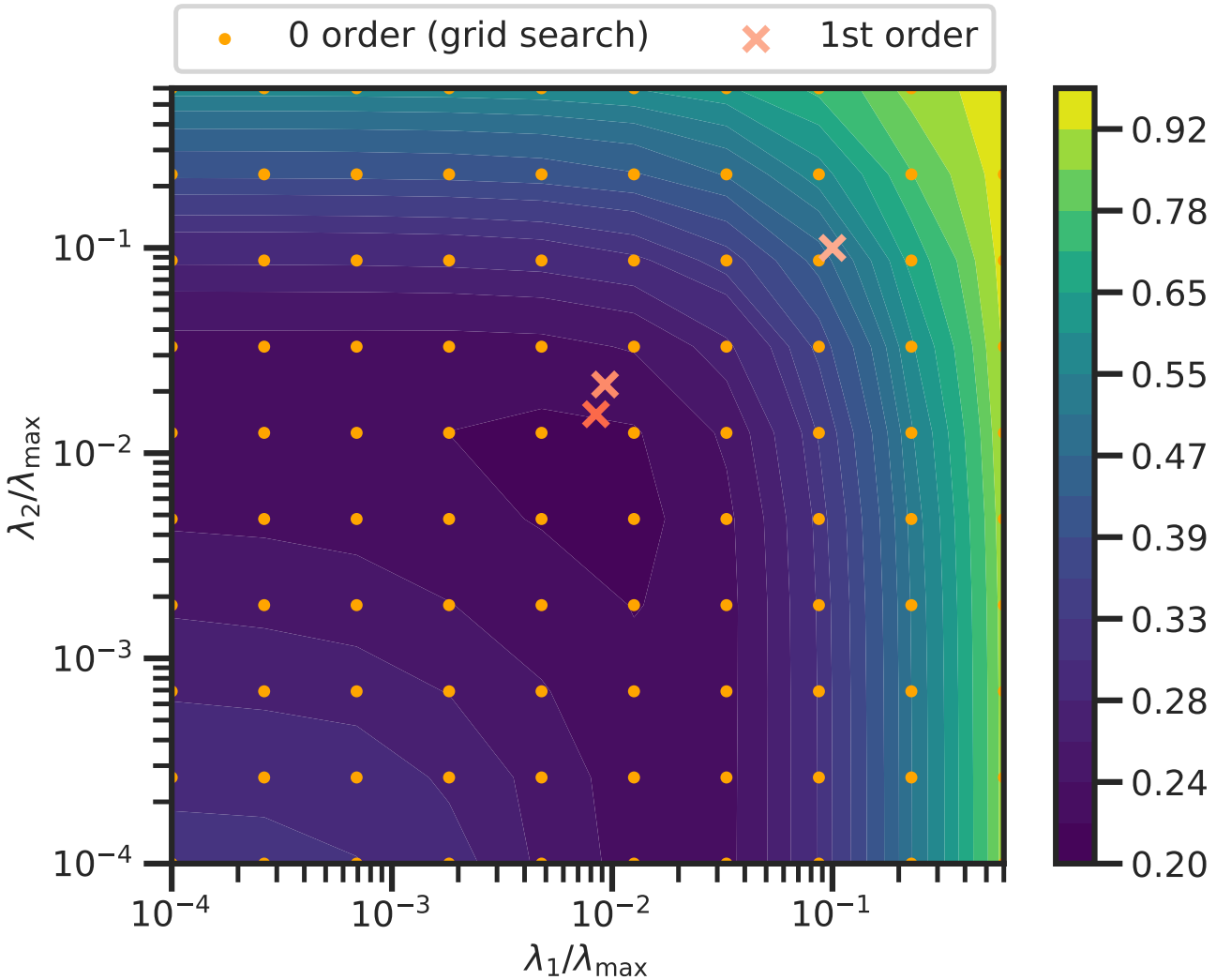
First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

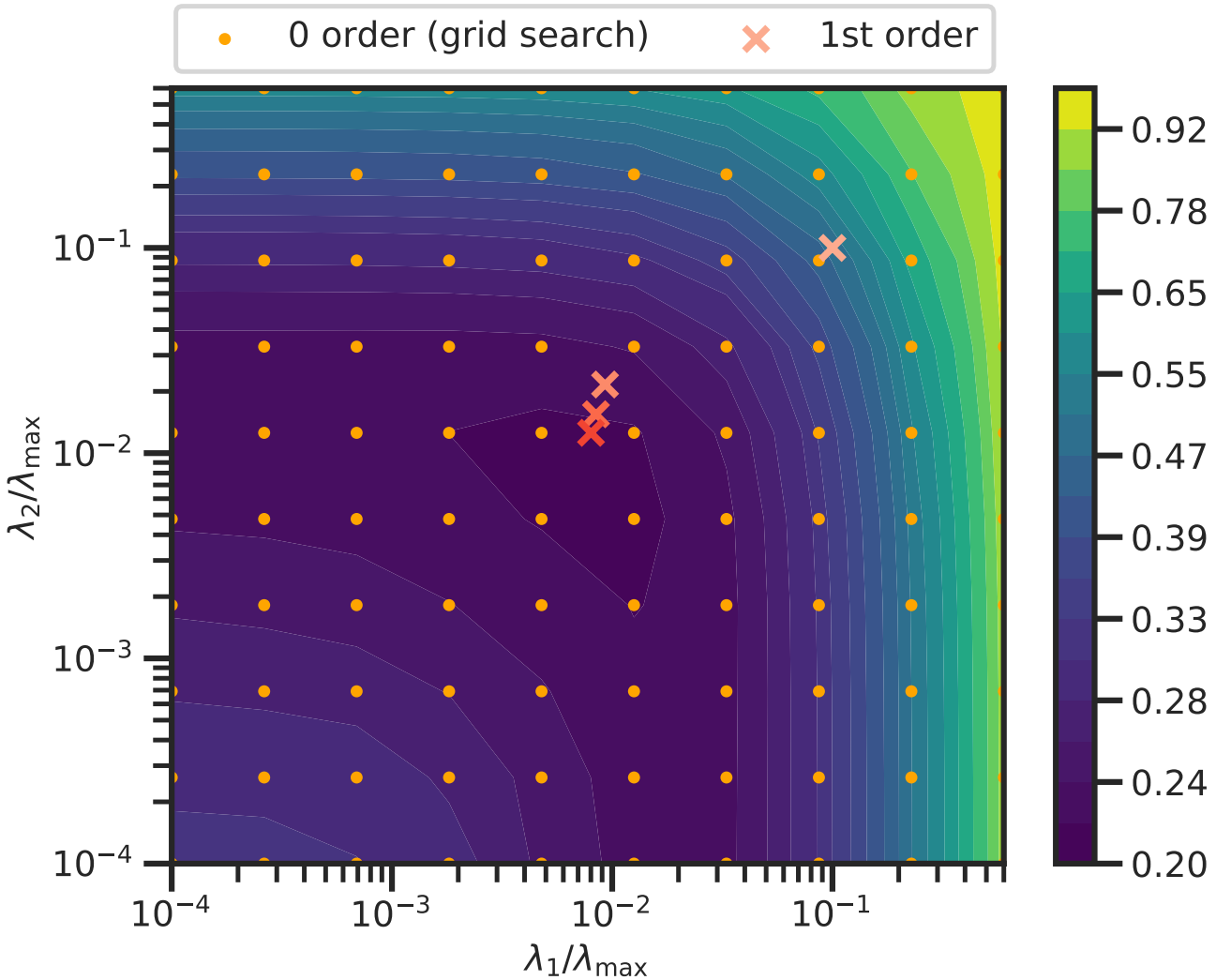
First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

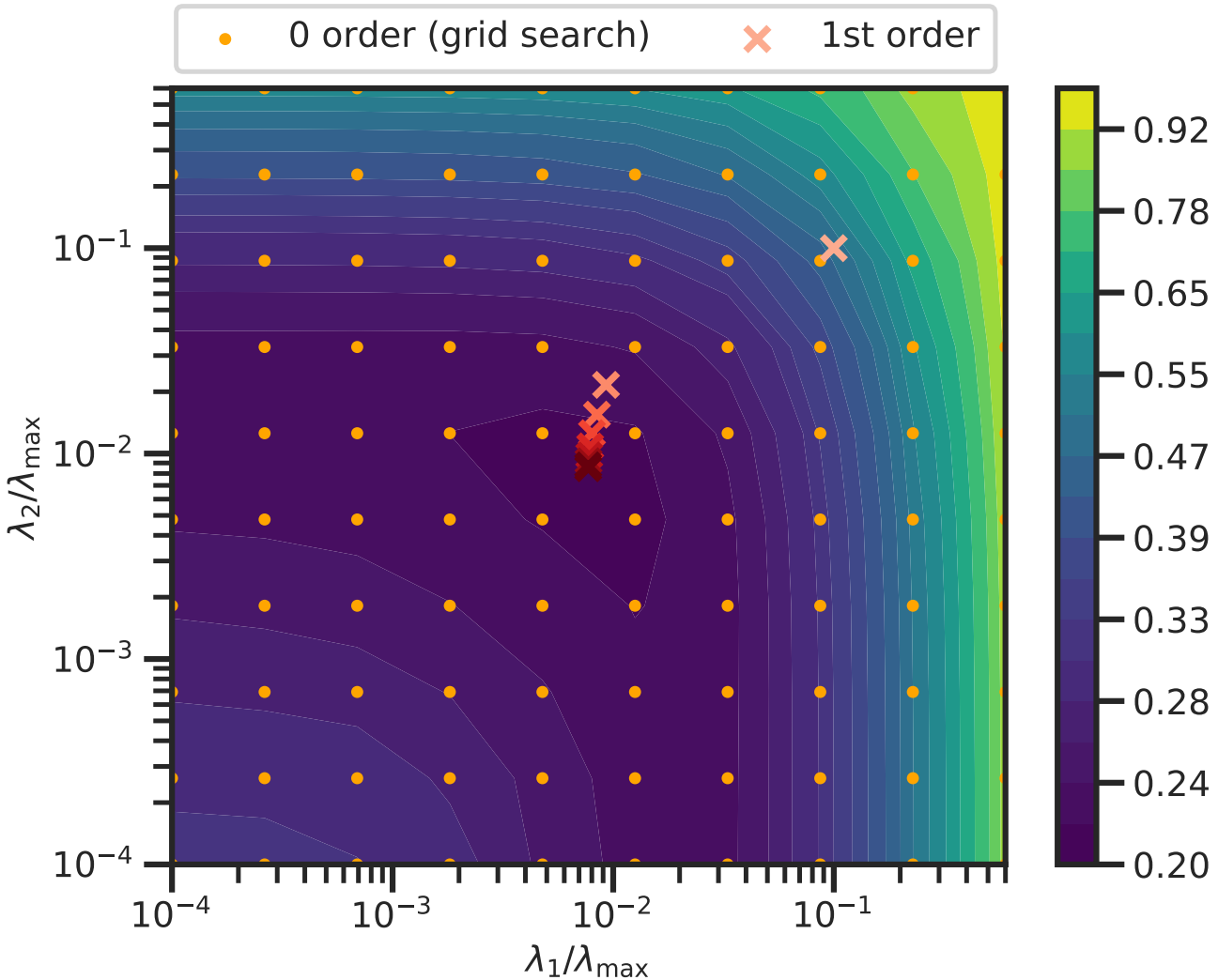
First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

First order optimization in λ, Enet



Real-sim dataset, level sets of the validation loss (held out)

$$\arg\min_\beta \frac{1}{2n}\|\mathbf{y}^{\text{train}} - X^{\text{train}}\beta\|^2 + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|^2$$

What's hard? Computing $\nabla_\lambda \mathcal{L}(\lambda)$

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq C(\hat{\beta}^{(\lambda)}) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

Once $\nabla_\lambda \mathcal{L}(\lambda)$ is computed, let's pretend "life is easy":

- Line-search[31]
- LBFGS[32]
- Gradient descent

---

[31] J. Nocedal and S. J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006, Chap. 3.

[32] D. Goldfarb. "A family of variable-metric methods derived by variational means". In: *Mathematics of computation* 24.109 (1970), pp. 23–26.

What's hard? Computing $\nabla_\lambda \mathcal{L}(\lambda)$

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \stackrel{\Delta}{=} C(\hat{\beta}^{(\lambda)}) \stackrel{\Delta}{=} \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

Once $\nabla_\lambda \mathcal{L}(\lambda)$ is computed, let's pretend "life is easy":

- Line-search[31]
- LBFGS[32]
- Gradient descent

The main challenge is to compute $\nabla_\lambda \mathcal{L}(\lambda)$ for a given λ!

---

[31] J. Nocedal and S. J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006, Chap. 3.

[32] D. Goldfarb. "A family of variable-metric methods derived by variational means". In: *Mathematics of computation* 24.109 (1970), pp. 23–26.

What's hard? Computing $\nabla_\lambda \mathcal{L}(\lambda)$

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq C(\hat{\beta}^{(\lambda)}) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

Once $\nabla_\lambda \mathcal{L}(\lambda)$ is computed, let's pretend "life is easy":

- Line-search[31]
- LBFGS[32]
- Gradient descent

The main challenge is to compute $\nabla_\lambda \mathcal{L}(\lambda)$ for a given λ!

---

[31] J. Nocedal and S. J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006, Chap. 3.

[32] D. Goldfarb. "A family of variable-metric methods derived by variational means". In: *Mathematics of computation* 24.109 (1970), pp. 23–26.

How to compute $\nabla_\lambda \mathcal{L}(\lambda)$?

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq C(\hat{\beta}^{(\lambda)}) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

Chain rule and Jacobian:

$$\nabla_\lambda \mathcal{L}(\lambda) = \underbrace{\hat{\mathcal{J}}_{(\lambda)}^\top}_{\triangleq (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \ldots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})} \nabla_\beta C(\hat{\beta}^{(\lambda)})$$

$\rightarrow$main challenge

- Q: How to compute the Jacobian $\hat{\mathcal{J}}_{(\lambda)}$ efficiently?
  A: Use prox properties and sparsity of the solution

How to compute $\nabla_\lambda \mathcal{L}(\lambda)$?

$$\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) \triangleq C(\hat{\beta}^{(\lambda)}) \triangleq \|\mathbf{y}^{\text{test}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}$$

$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|\mathbf{y}^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1$$

Chain rule and Jacobian:

$$\nabla_\lambda \mathcal{L}(\lambda) = \underbrace{\hat{\mathcal{J}}_{(\lambda)}^\top}_{\triangleq (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \ldots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})} \nabla_\beta C(\hat{\beta}^{(\lambda)})$$

$\rightarrow$main challenge

- **Q: How to compute the Jacobian $\hat{\mathcal{J}}_{(\lambda)}$ efficiently?**
  A: Use prox properties and sparsity of the solution

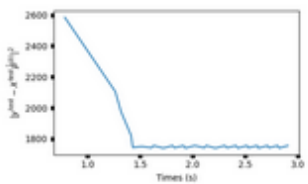
sparse-ho

https://qb3.github.io/sparse-ho/

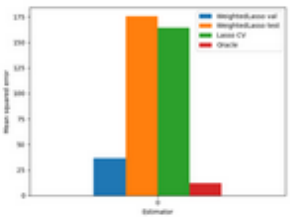
sparse-ho 0.1.dev Examples API GitHub Site

Examples Gallery

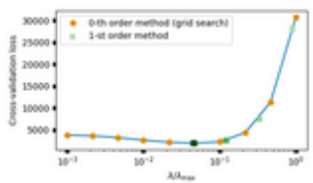
The examples gallery provides working code samples demonstrating what can be done with the sparse_ho library.



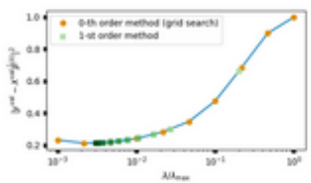
Monitor custom metrics along hyperoptimization



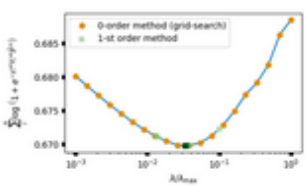
Weighted Lasso with held-out test set



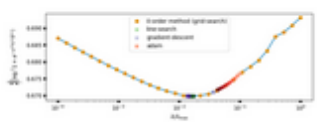
Lasso with Cross-validation



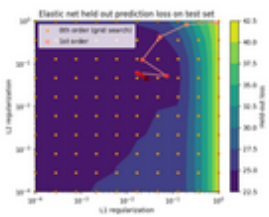
Lasso with held-out test set



Sparse logistic regression



Compare outer optimizers



Elastic net with held-out test set

Outline

Outline

Structured support

Here we suppose that we have a known group structure on the variables (prior the experiment) : $[\![1, p]\!] = \bigcup_{g \in \mathcal{G}} g$

Vector and active coordinate (in orange):

Sparse support: any

Possible penalties: Lasso

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\boldsymbol{\beta}_j|$$

Structured support

Here we suppose that we have a known group structure on the variables (prior the experiment) : $[\![1, p]\!] = \bigcup_{g \in \mathcal{G}} g$

Vector and active coordinate (in orange):

Sparse support: group

Possible penalties: Group-Lasso

$$\|\boldsymbol{\beta}\|_{2,1} = \sum_{g \in G} \|\boldsymbol{\beta}_g\|_2$$

Structured support

Here we suppose that we have a known group structure on the variables (prior the experiment) : $[\![1, p]\!] = \bigcup_{g \in \mathcal{G}} g$

Vector and active coordinate (in orange):

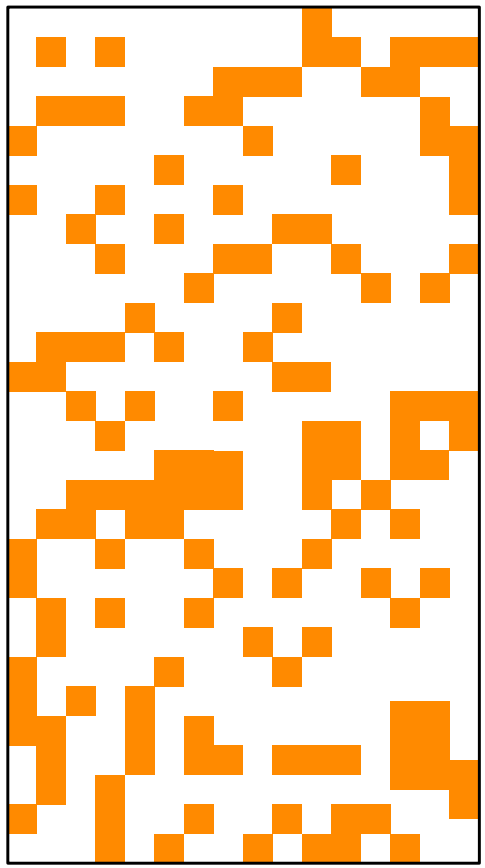
Sparse support: group + sub-groups

Possible penalties: Sparse-Group-Lasso

$$\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_{2,1} = \alpha \sum_{j=1}^{p} |\boldsymbol{\beta}_j| + (1-\alpha) \sum_{g \in G} \|\boldsymbol{\beta}_g\|_2$$

Multi-tasks penalties

Penalties need to be adapted for matrix regularization:



Parameter $\mathrm{B} \in \mathbb{R}^{p \times q}$

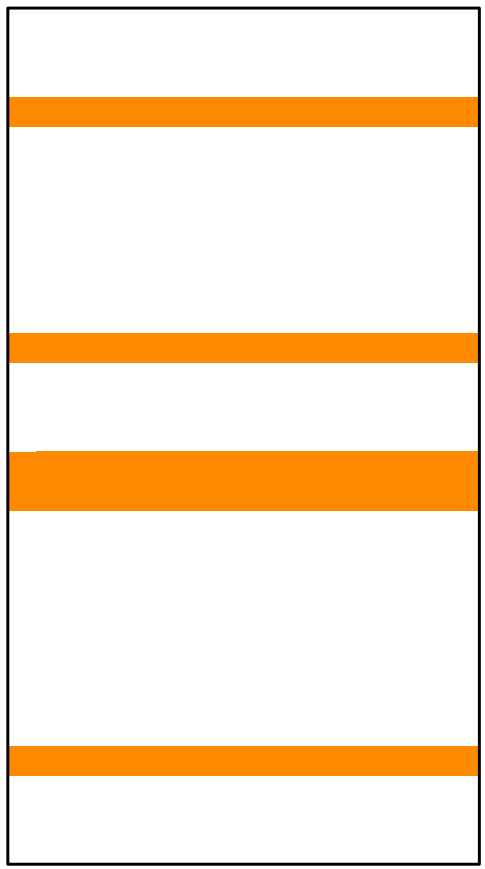
Sparse support:
any

Penalty: Lasso

$$\|\mathrm{B}\|_1 = \sum_{j=1}^{p} \sum_{k=1}^{q} |\mathrm{B}_{j,k}|$$

Multi-tasks penalties

Penalties need to be adapted for matrix regularization:



Parameter $\mathrm{B} \in \mathbb{R}^{p \times q}$

Sparse support:
group

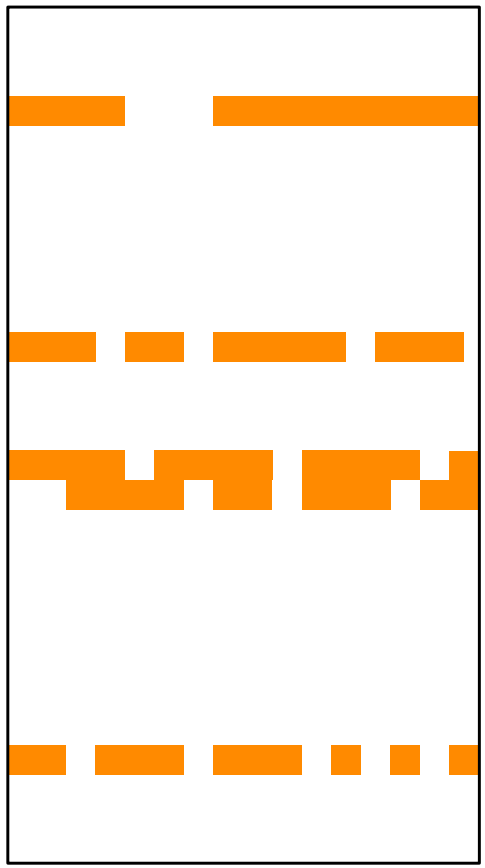
Penalty: Group-Lasso

$$\|\mathrm{B}\|_{2,1} = \sum_{j=1}^{p} \|\mathrm{B}_{j:}\|_2$$

where $\mathrm{B}_{j,:}$ the j-th line of B

Multi-tasks penalties

Penalties need to be adapted for matrix regularization:



Parameter $\mathrm{B} \in \mathbb{R}^{p \times q}$

Sparse support:
group + sub-groups

Penalty: Sparse-Group-Lasso

$$\alpha \|\mathrm{B}\|_1 + (1-\alpha)\|\mathrm{B}\|_{2,1}$$

Conclusion

- convex optimization for sparse inverse / learning problem
- efficient solvers for convex case (non-convex wilder)
- code importance for applied field (and parameter tuning)

More details on `sparse-ho`:
Quentin Bertrand at *Séminaires EPS* (IMAG), May 03 (13:45)

Powered with **MooseTeX**

Bibliographie I

- Bauschke, H. H. and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. New York: Springer, 2011, pp. xvi+468.
- Beck, A. and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM J. Imaging Sci.* 2.1 (2009), pp. 183–202.
- Bergstra, J. and Y. Bengio. "Random search for hyper-parameter optimization". In: *J. Mach. Learn. Res.* 13.2 (2012).
- Bertsimas, D., A. King, and R. Mazumder. "Best subset selection via a modern optimization lens". In: *Ann. Statist.* 44.2 (2016), pp. 813–852.
- Bessière, C. et al. "Probing instructions for expression regulation in gene nucleotide compositions". In: *PLOS Computational Biology* 14.1 (Jan. 2018), pp. 1–28.
- Bonnefoy, A. et al. "A dynamic screening principle for the lasso". In: *EUSIPCO*. 2014, pp. 6–10.

Bibliographie II

- Boyd, S. et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.
- Brochu, E., V. M. Cora, and N. De Freitas. “A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning”. In: (2010).
- Bunea, F., A. B. Tsybakov, and M. H. Wegkamp. “Aggregation for Gaussian regression”. In: *Ann. Statist.* 35.4 (2007), pp. 1674–1697.
- Donoho, D. L., A., and A. Montanari. “Message-passing algorithms for compressed sensing”. In: *Proceedings of the National Academy of Sciences* 106.45 (2009), pp. 18914–18919.
- Efron, B. et al. “Least angle regression”. In: *Ann. Statist.* 32.2 (2004). With discussion, and a rejoinder by the authors, pp. 407–499.

Bibliographie III

- Efroymson, M. A. "Multiple regression analysis". In: *Mathematical methods for digital computers*. New York: Wiley, 1960, pp. 191–203.
- El Ghaoui, L., V. Viallon, and T. Rabbani. "Safe feature elimination in sparse supervised learning". In: *J. Pacific Optim.* 8.4 (2012), pp. 667–698.
- Friedman, J. et al. "Pathwise coordinate optimization". In: *Ann. Appl. Stat.* 1.2 (2007), pp. 302–332.
- Goldfarb, D. "A family of variable-metric methods derived by variational means". In: *Mathematics of computation* 24.109 (1970), pp. 23–26.
- Golub, T. R. et al. "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.". In: *Science* 286.5439 (1999), pp. 531–537.

Bibliographie IV

- Johnson, T. B. and C. Guestrin. "Blitz: A Principled Meta-Algorithm for Scaling Sparse Optimization". In: *ICML*. Vol. 37. 2015, pp. 1171–1179.
- Lange, K. *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016, pp. ix+223.
- Mairal, J. and B. Yu. "Complexity analysis of the Lasso regularization path". In: *ICML*. 2012, pp. 353–360.
- Mallat, S. and Z. Zhang. "Matching Pursuit With Time-Frequency Dictionaries". In: *IEEE Trans. Image Process.* 41 (1993), pp. 3397–3415.
- Martinet, B. "Brève communication. Régularisation d'inéquations variationnelles par approximations successives". In: *Revue française d'informatique et de recherche opérationnelle. Série rouge* 4.R3 (1970), pp. 154–158.

Bibliographie V

- Moreau, J.-J. “Fonctions convexes duales et points proximaux dans un espace hilbertien”. In: *C. R. Acad. Sci. Paris* 255 (1962), pp. 2897–2899.
- Natarajan, B. K. “Sparse approximate solutions to linear systems”. In: *SIAM J. Comput.* 24.2 (1995), pp. 227–234.
- Ndiaye, E. et al. “Safe Grid Search with Optimal Complexity”. In: *ICML*. Vol. 97. 2019, pp. 4771–4780.
- Nesterov, Y. “A method for solving a convex programming problem with rate of convergence $O(1/k^2)$”. In: *Soviet Math. Doklady* 269.3 (1983), pp. 543–547.
- Nocedal, J. and S. J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006.
- Ochs, P. et al. “On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision”. In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

Bibliographie VI

- Osborne, M. R., B. Presnell, and B. A. Turlach. "A new approach to variable selection in least squares problems". In: *IMA J. Numer. Anal.* 20.3 (2000), pp. 389–403.
- Pedregosa, F. "Hyperparameter optimization with approximate gradient". In: *ICML*. Vol. 48. 2016, pp. 737–746.
- Tibshirani, R. "Regression Shrinkage and Selection via the Lasso". In: *J. R. Stat. Soc. Ser. B Stat. Methodol.* 58.1 (1996), pp. 267–288.
- Tseng, P. "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *J. Optim. Theory Appl.* 109.3 (2001), pp. 475–494.
- Tsybakov, A. B. "Optimal Rates of Aggregation". In: *COLT*. 2003, pp. 303–313.
- Zhang, T. "Adaptive forward-backward greedy algorithm for learning sparse representations". In: *IEEE Trans. Inf. Theory* 57.7 (2011), pp. 4689–4708.

Proof (can be skipped)

Quadratic majorization

If f is convex, differentiable with gradient L-Lipschitz, *i.e.*,

$$\forall (\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\boldsymbol{\beta}) - \nabla f(\boldsymbol{\beta}')\| \leq L \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|$$

then the following holds: $\forall (\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p,$

$$0 \leq f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}') - \langle \nabla f(\boldsymbol{\beta}'), \boldsymbol{\beta} - \boldsymbol{\beta}' \rangle \leq \frac{L}{2} \|\boldsymbol{\beta}' - \boldsymbol{\beta}\|^2$$

Proof (can be skipped)

Quadratic majorization

If f is convex, differentiable with gradient L-Lipschitz, *i.e.*,

$$\forall(\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\boldsymbol{\beta}) - \nabla f(\boldsymbol{\beta}')\| \leq L\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|$$

then the following holds: $\forall(\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p,$

$$0 \leq f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}') - \langle \nabla f(\boldsymbol{\beta}'), \boldsymbol{\beta} - \boldsymbol{\beta}' \rangle \leq \frac{L}{2} \|\boldsymbol{\beta}' - \boldsymbol{\beta}\|^2$$

a) 1st inequality : consequence of convexity

b) 2nd inequality: Taylor expansion

Proof (can be skipped)

Quadratic majorization

If f is convex, differentiable with gradient L-Lipschitz, *i.e.*,

$$\forall(\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p, \quad \|\nabla f(\boldsymbol{\beta}) - \nabla f(\boldsymbol{\beta}')\| \leq L\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|$$

then the following holds: $\forall(\boldsymbol{\beta}, \boldsymbol{\beta}') \in \mathbb{R}^p \times \mathbb{R}^p,$

$$0 \leq f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}') - \langle \nabla f(\boldsymbol{\beta}'), \boldsymbol{\beta} - \boldsymbol{\beta}' \rangle \leq \frac{L}{2} \|\boldsymbol{\beta}' - \boldsymbol{\beta}\|^2$$

a) 1st inequality : consequence of convexity

b) 2nd inequality: Taylor expansion

Rem: if f is twice differentiable $\nabla^2 f \preceq L \cdot \mathrm{Id}_d \iff L \cdot \mathrm{Id}_d - \nabla^2 f$ is semi-definite positive, then ∇f is L-Lipschitz

Proof (can be skipped)

Fix $\boldsymbol{\beta}^0$, and assume the previous inequality holds for any $\boldsymbol{\beta} \in \mathbb{R}^p$:

$$f(\boldsymbol{\beta}) - f(\boldsymbol{\beta}^0) - \langle \nabla f(\boldsymbol{\beta}^0), \boldsymbol{\beta} - \boldsymbol{\beta}^0 \rangle \leq \frac{L}{2} \left\| \boldsymbol{\beta}^0 - \boldsymbol{\beta} \right\|^2$$

this yields

$$\begin{aligned} f(\boldsymbol{\beta}) \leq & f(\boldsymbol{\beta}^0) + \langle \nabla f(\boldsymbol{\beta}^0), \boldsymbol{\beta} - \boldsymbol{\beta}^0 \rangle + \frac{L}{2} \|\boldsymbol{\beta}^0 - \boldsymbol{\beta}\|^2 \\ = & \tfrac{L}{2} \left\| \boldsymbol{\beta}^0 - \tfrac{1}{L} \nabla f(\boldsymbol{\beta}^0) - \boldsymbol{\beta} \right\|^2 + f(\boldsymbol{\beta}^0) - \tfrac{1}{2L} \left\| \nabla f(\boldsymbol{\beta}^0) \right\|^2 \\ := & g(\boldsymbol{\beta}^0, \boldsymbol{\beta}) \end{aligned}$$

Hence : $\quad \forall \boldsymbol{\beta} \in \mathbb{R}^p, \quad \begin{cases} g(\boldsymbol{\beta}^0 | \boldsymbol{\beta}^0) = f(\boldsymbol{\beta}^0) \\ f(\boldsymbol{\beta}) \leq g(\boldsymbol{\beta}^0 | \boldsymbol{\beta}) \end{cases}$

$\Longrightarrow$ tight upper bound that can be minimized:

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min}\, g(\boldsymbol{\beta}^0 | \boldsymbol{\beta}) = \boldsymbol{\beta}^0 - \frac{1}{L} \nabla f(\boldsymbol{\beta}^0)$$