Implicit differentiation for fast hyperparameter selection in non-smooth convex learning

Joseph Salmon (IMAG, Univ. Montpellier, CNRS)

Contact:

joseph.salmon@umontpellier.fr

http://josephsalmon.eu

Github: @josephsalmon

Twitter: @salmonjsph

Implicit differentiation for fast hyperparameter selection ~~in non-smooth convex learning~~ for Lasso and co

Joseph Salmon (IMAG, Univ. Montpellier, CNRS)

Contact:

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)

Alexandre Gramfort (INRIA, Parietal Team)

Quentin B.

Quentin K.

Mathieu

Samuel

Mathurin

Alexandre

Looking for post-doc in ML? Montpellier?

Motivation

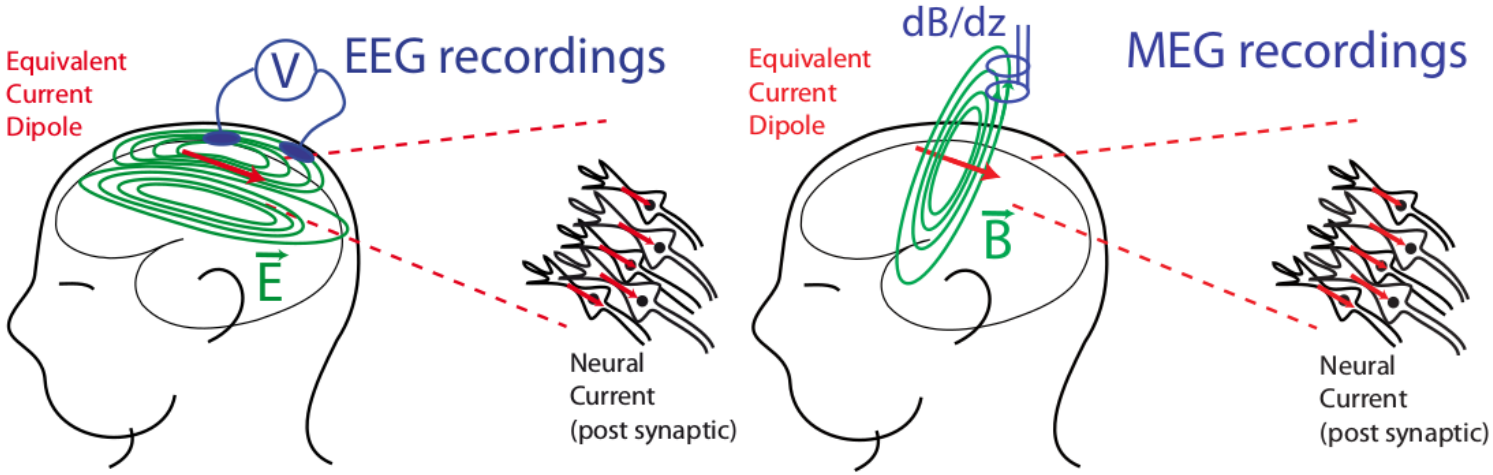
Hyperparameter optimization

Hypergradient computation

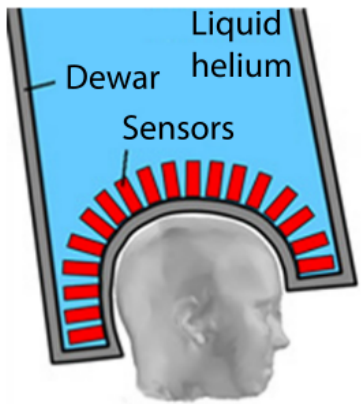
Experiments

M/EEG inverse problem for brain imaging

- sensors: electric and magnetic fields during a cognitive task
- goal: which parts of the brain are responsible for the signals?
- applications: epilepsy treatment, brain aging, anesthesia risks

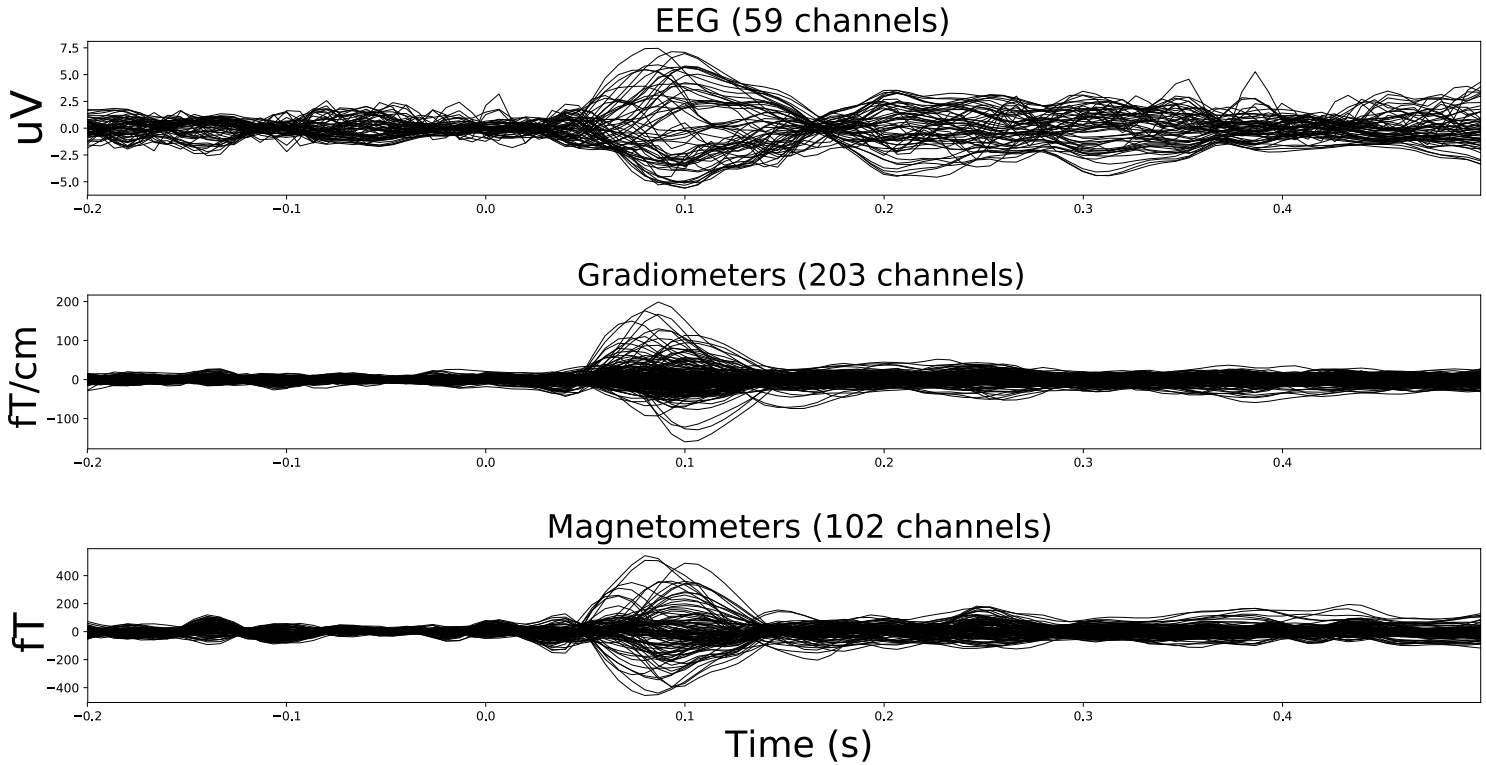


First EEG recordings in 1929 by H. Berger



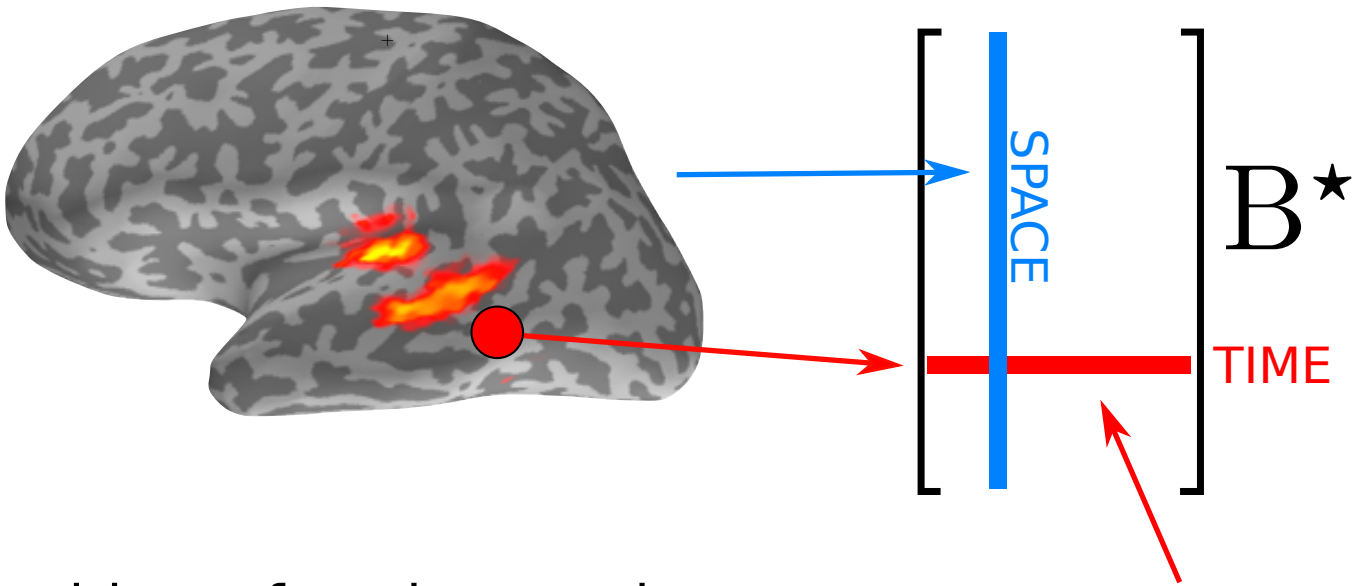
Hôpital La Timone
Marseille, France

M/EEG data

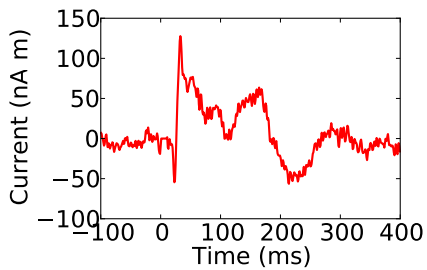


- 3 different types of sensor

Source modeling (discretization with voxels)

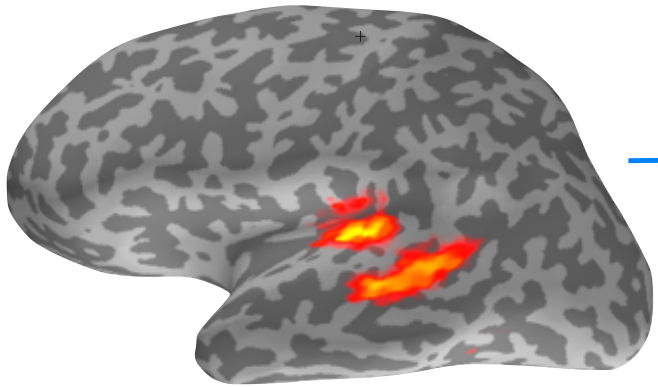


Position a few thousands candidate sources over the brain (*e.g.*, every 5mm)



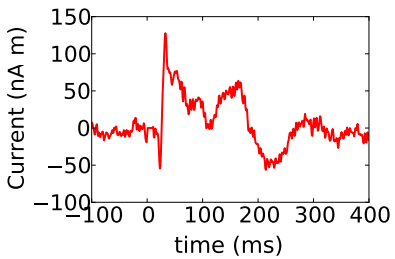
$$\mathrm{B}^\star \in \mathbb{R}^{p \times T}$$

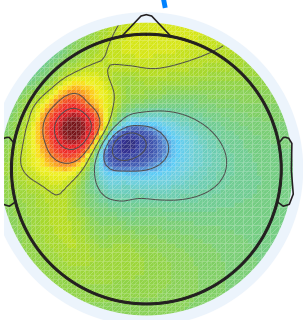
The M/EEG inverse problem: modeling

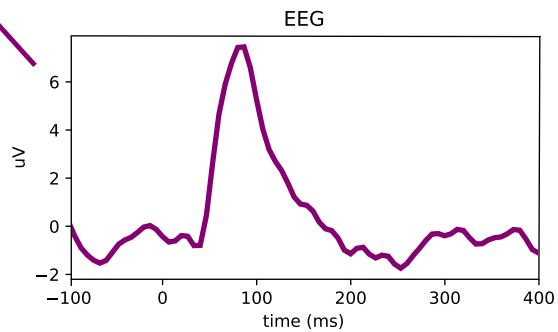


$$T \approx 100 \qquad p \approx 10000$$

$$n \approx 100 \qquad \begin{bmatrix} Y \end{bmatrix} = \begin{bmatrix} \end{bmatrix} \begin{bmatrix} X \end{bmatrix} \begin{bmatrix} B* \end{bmatrix} + E$$





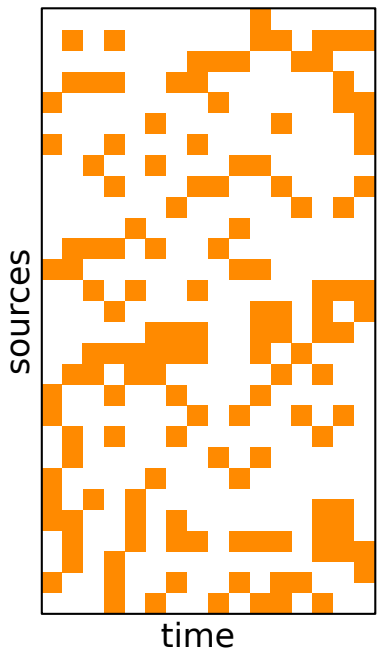


$$n \ll p$$

Multi-Task penalties[1]

Popular convex penalties considered:

$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \Omega(\mathrm{B}) \right)$$



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

Sparse support: no structure

Penalty: **Lasso type**

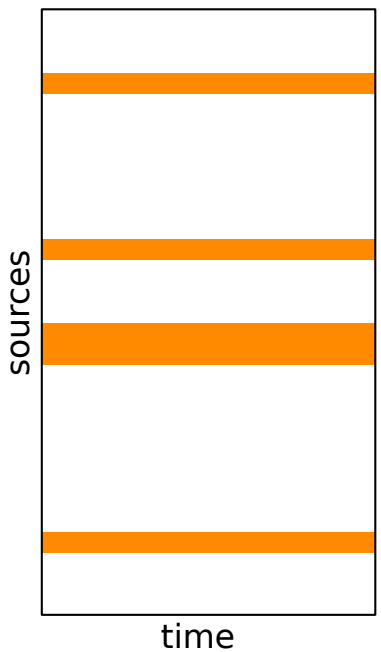
$$\Omega(\mathrm{B}) = \|\mathrm{B}\|_1 = \sum_{j=1}^{p} \sum_{k=1}^{q} |\mathrm{B}_{j,k}|$$

[1] G. Obozinski, B. Taskar, and M. I. Jordan. "Joint covariate selection and joint subspace selection for multiple classification problems". In: Statistics and Computing 20.2 (2010), pp. 231–252.

Multi-Task penalties[1]

Popular convex penalties considered: Multi-Task Lasso (MTL)

$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \Omega(\mathrm{B}) \right)$$



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

Sparse support: group structure ✓

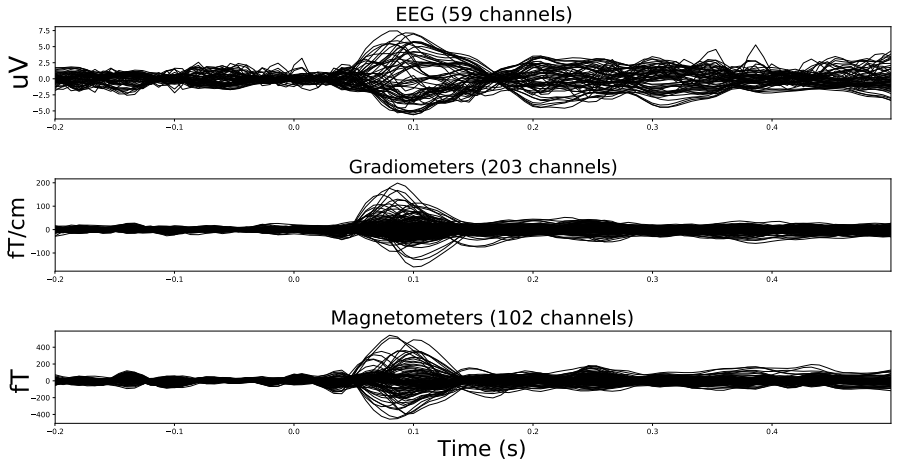
Penalty: **Group-Lasso type**

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

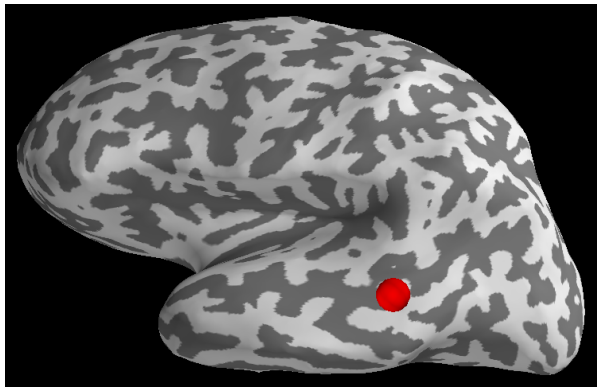
where $\mathrm{B}_{j,:}$ the j-th line of B

[1] G. Obozinski, B. Taskar, and M. I. Jordan. "Joint covariate selection and joint subspace selection for multiple classification problems". In: Statistics and Computing 20.2 (2010), pp. 231–252.

Summary



What you have: Y



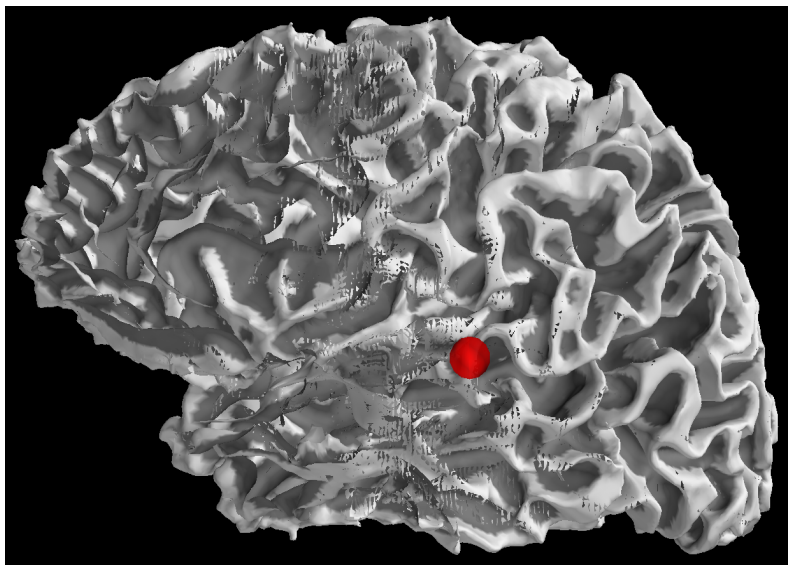
What you want: B

This is typically done using optimization based estimators:

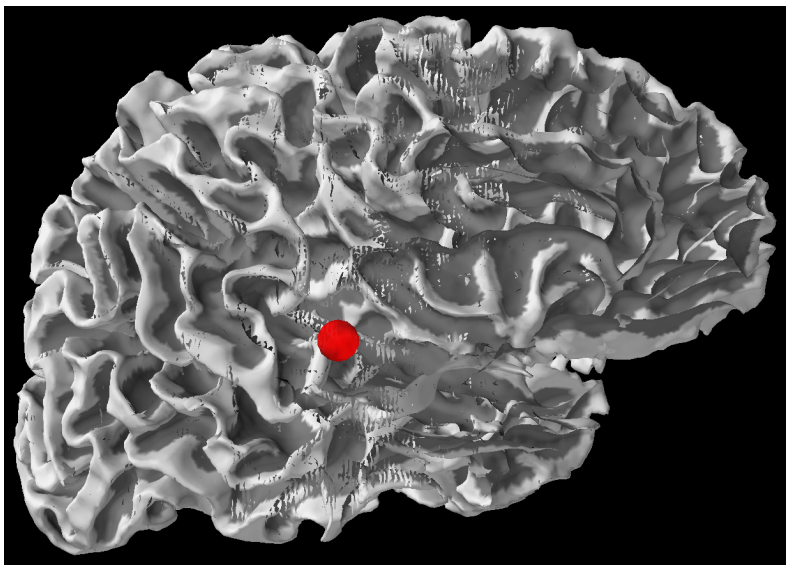
$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \Omega(\mathrm{B}) \right)$$

Which λ to pick?

$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \|\mathrm{B}\|_{2,1} \right)$$



Left hemisphere: $\lambda = 0.8\lambda_{\max}$

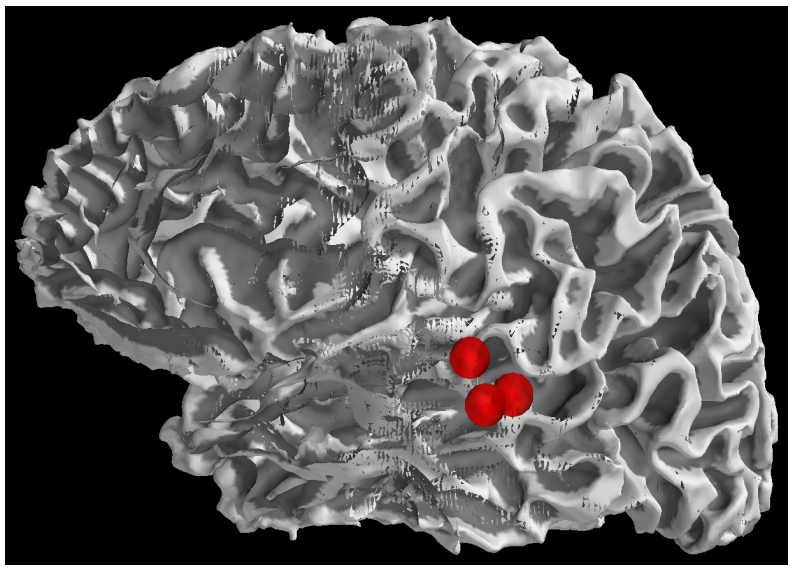


Right hemisphere: $\lambda = 0.8\lambda_{\max}$

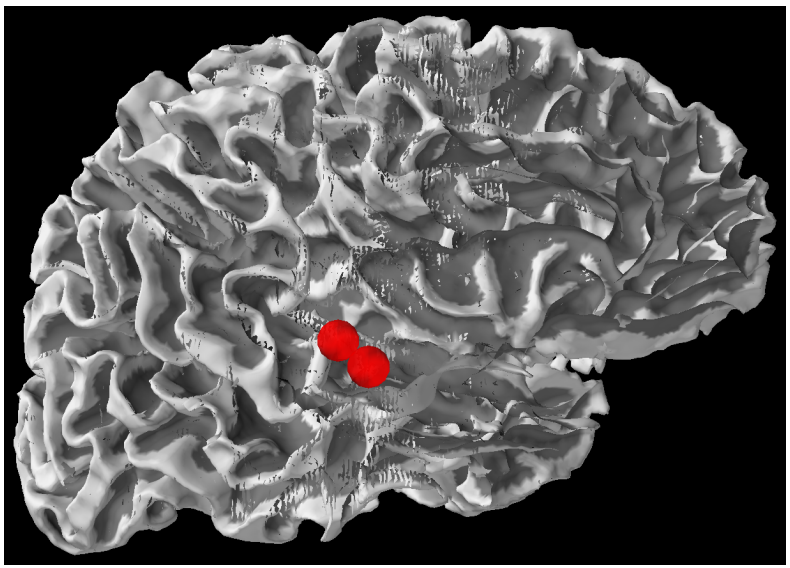
- When $\lambda \geq \lambda_{\max}$, $\hat{\mathrm{B}} = 0$ no sources are recovered

Which λ to pick?

$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \|\mathrm{B}\|_{2,1} \right)$$



Left hemisphere: $\lambda = 0.6\lambda_{\max}$

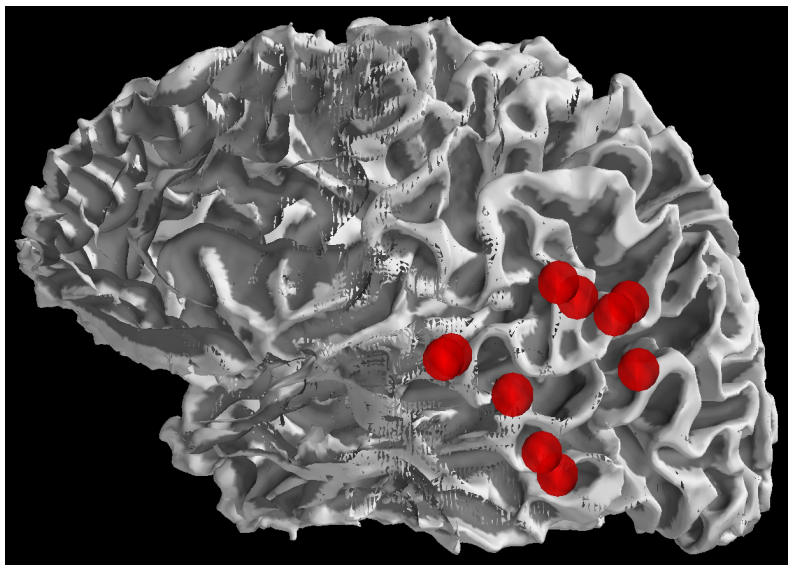


Right hemisphere: $\lambda = 0.6\lambda_{\max}$

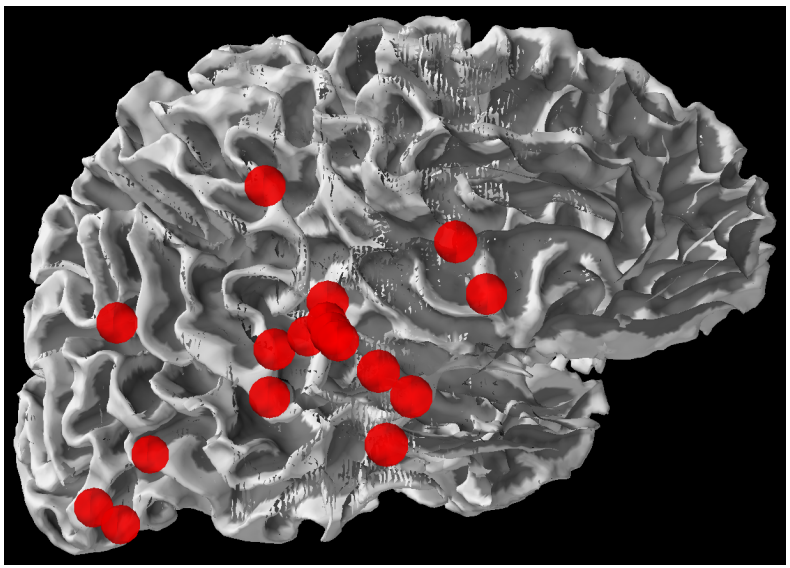
- When $\lambda \geq \lambda_{\max}$, $\hat{\mathrm{B}} = 0$ no sources are recovered

Which λ to pick?

$$\hat{\mathrm{B}} \in \underset{\mathrm{B} \in \mathbb{R}^{p \times T}}{\arg\min} \left(\frac{1}{2nT} \|Y - X\mathrm{B}\|_F^2 + \lambda \|\mathrm{B}\|_{2,1} \right)$$



Left hemisphere: $\lambda = 0.1\lambda_{\max}$



Right hemisphere: $\lambda = 0.1\lambda_{\max}$

- When $\lambda \geq \lambda_{\max}$, $\hat{\mathrm{B}} = 0$ no sources are recovered

Which λ to pick? A statistical persective[2] (i.i.d. case, Single-Task, $y = X\beta + \sigma^* \varepsilon$)

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

Theorem

- i.i.d. Gaussian noise
- $+$ X satisfying the "Restricted Eigenvalue" property
- $+$ $\lambda = 2\sigma_* \sqrt{\frac{2 \log(p/\delta)}{n}}$
- $\Longrightarrow$ with probability $1 - \delta$:

$$\frac{1}{n} \|X\beta^* - X\hat{\beta}^{(\lambda)}\|^2 \leq \frac{18}{\kappa_{s^*}^2} \frac{\sigma_*^2 s^*}{n} \log\left(\frac{p}{\delta}\right)$$

BUT σ_* is unknown in practice !

[2] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. "Simultaneous analysis of Lasso and Dantzig selector". In: Ann. Statist. 37.4 (2009), pp. 1705–1732.

Which λ to pick? A statistical persective[2] (i.i.d. case, Single-Task, $y = X\beta + \sigma^* \varepsilon$)

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

Theorem

- i.i.d. Gaussian noise
- $+$ X satisfying the "Restricted Eigenvalue" property
- $+$ $\lambda = 2\sigma_* \sqrt{\frac{2 \log(p/\delta)}{n}}$
- $\Longrightarrow$ with probability $1 - \delta$:

$$\frac{1}{n} \|X\beta^* - X\hat{\beta}^{(\lambda)}\|^2 \leq \frac{18}{\kappa_{s^*}^2} \frac{\sigma_*^2 s^*}{n} \log\left(\frac{p}{\delta}\right)$$

BUT σ_* is unknown in practice !

[2] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. "Simultaneous analysis of Lasso and Dantzig selector". In: Ann. Statist. 37.4 (2009), pp. 1705–1732.

Which λ to pick? A statistical persective II[3] (i.i.d. case, Single-Task)

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{\sqrt{n}} \|y - X\beta\|_2 + \lambda \|\beta\|_1$$

Theorem

- i.i.d. Gaussian noise
- $+ X$ satisfying the "Restricted Eigenvalue" property
- $+ \lambda = 2\sqrt{\frac{2\log(p/\delta)}{n}}$
- $\Longrightarrow$ with probability $1 - \delta$:

$$\frac{1}{n}\|X\beta^* - X\hat{\beta}^{(\lambda)}\|^2 \leq \frac{18}{\kappa_{s^*}^2} \frac{\sigma_*^2 s^*}{n} \log\left(\frac{p}{\delta}\right)$$

λ does not depend on σ_* anymore!

[3] A. Belloni, V. Chernozhukov, and L. Wang. "Square-root Lasso: pivotal recovery of sparse signals via conic programming". In: Biometrika 98.4 (2011), pp. 791–806.

Which λ to pick? A statistical persective II[3] (i.i.d. case, Single-Task)

$$\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{\sqrt{n}} \|y - X\beta\|_2 + \lambda \|\beta\|_1$$

Theorem

- i.i.d. Gaussian noise
- $+ X$ satisfying the "Restricted Eigenvalue" property
- $+ \lambda = 2\sqrt{\frac{2\log(p/\delta)}{n}}$
- $\Longrightarrow$ with probability $1 - \delta$:

$$\frac{1}{n}\|X\beta^* - X\hat{\beta}^{(\lambda)}\|^2 \leq \frac{18}{\kappa_{s^*}^2} \frac{\sigma_*^2 s^*}{n} \log\left(\frac{p}{\delta}\right)$$

λ does not depend on σ_* anymore!

[3] A. Belloni, V. Chernozhukov, and L. Wang. "Square-root Lasso: pivotal recovery of sparse signals via conic programming". In: Biometrika 98.4 (2011), pp. 791–806.

Which λ to pick? A statistical persective III

- $\lambda \sim \sigma^*$ and λ independent of σ^* confirmed in practice ✓
- Strong statistical assumptions, not verified in practice ✗
- Still unknown quantities in the closed-form formula for λ: still needs calibration in practice ✗

Which λ to pick? A statistical persective III

- $\lambda \sim \sigma^*$ and λ independent of σ^* confirmed in practice ✓
- Strong statistical assumptions, not verified in practice ✗
- Still unknown quantities in the closed-form formula for λ: still needs calibration in practice ✗

Q: So how do practitioners proceed?

Which λ to pick? A statistical persective III

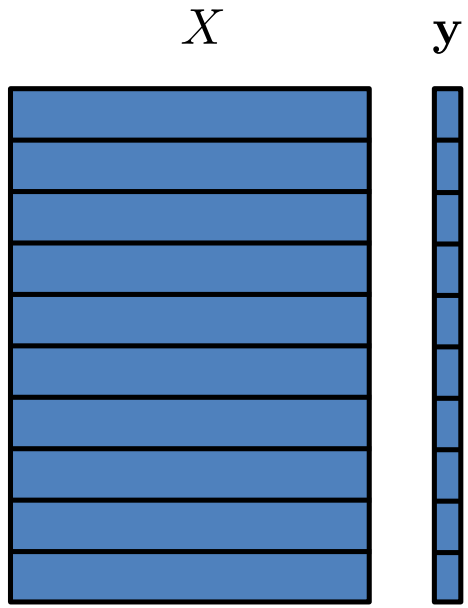
- $\lambda \sim \sigma^*$ and λ independent of σ^* confirmed in practice ✓
- Strong statistical assumptions, not verified in practice ✗
- Still unknown quantities in the closed-form formula for λ: still needs calibration in practice ✗

Q: So how do practitioners proceed?

R: Use cross-validation (see ML101)

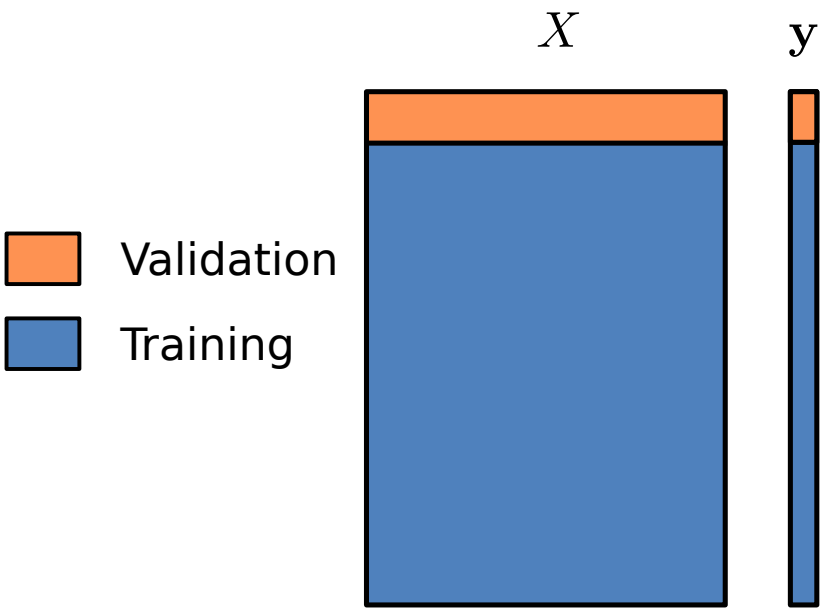
K-fold Cross-Validation($K = 10$)

- 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):

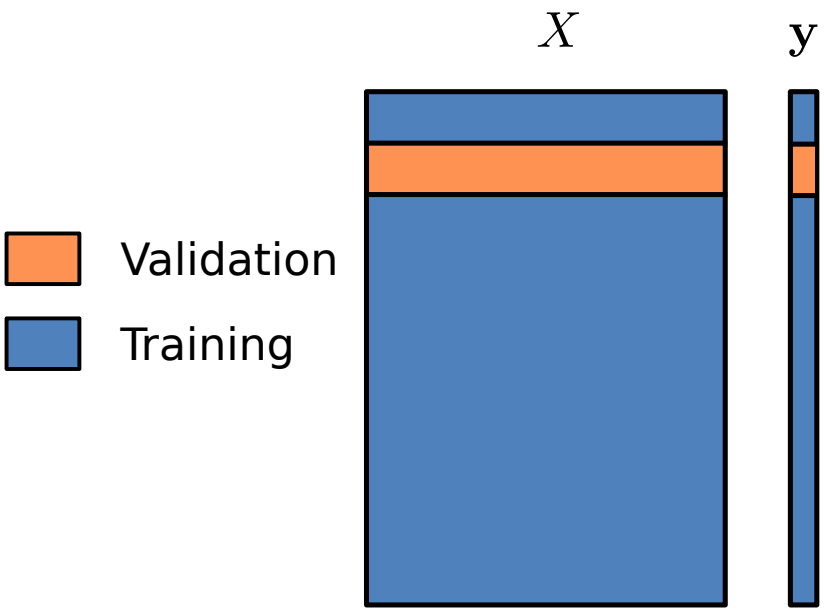


$$k = 1$$

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):

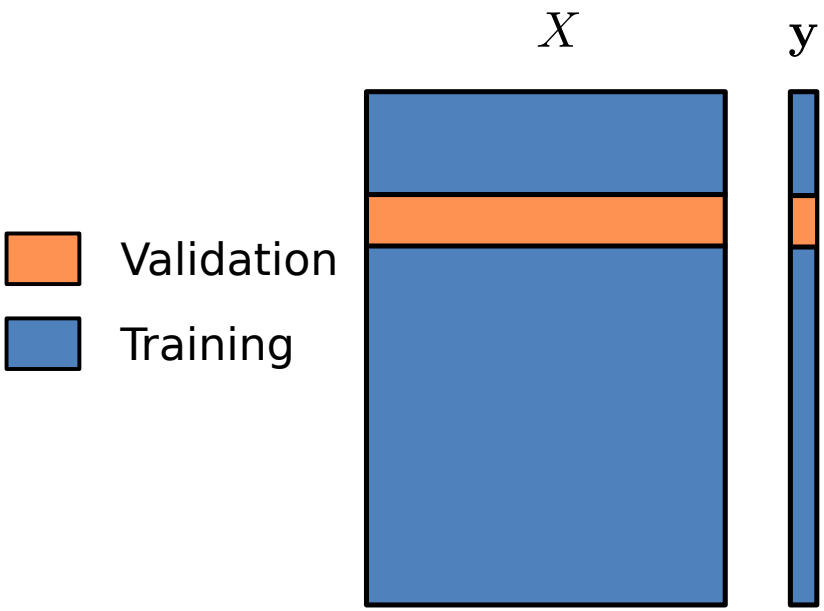


$$k = 2$$

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):

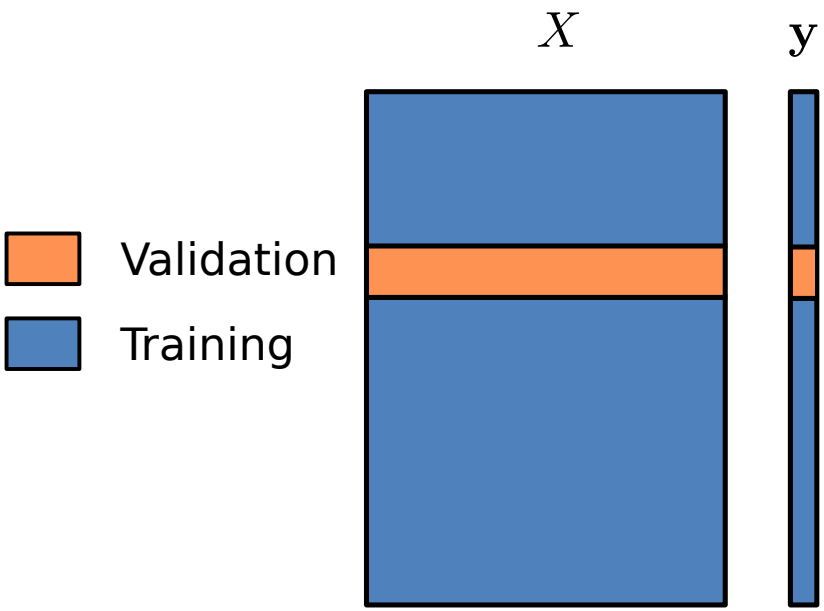


$$k = 3$$

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):

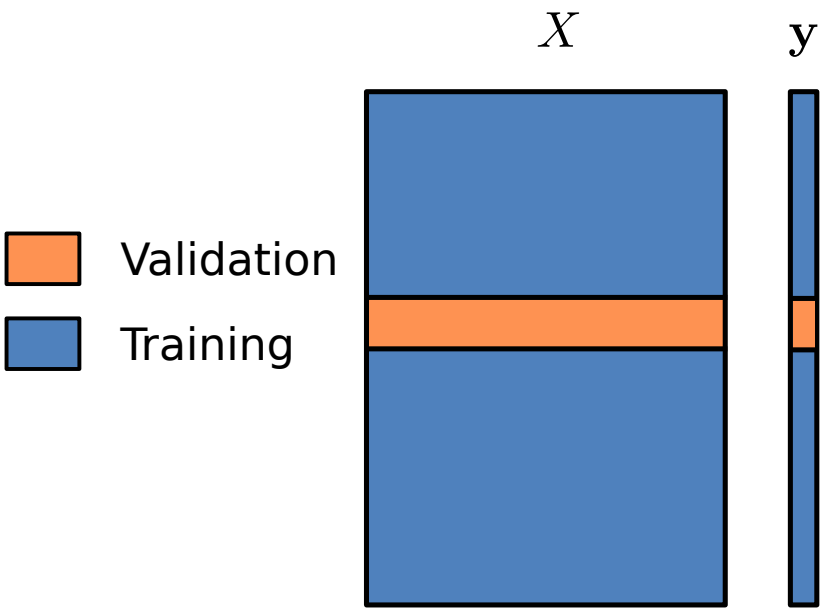


$$k = 4$$

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):

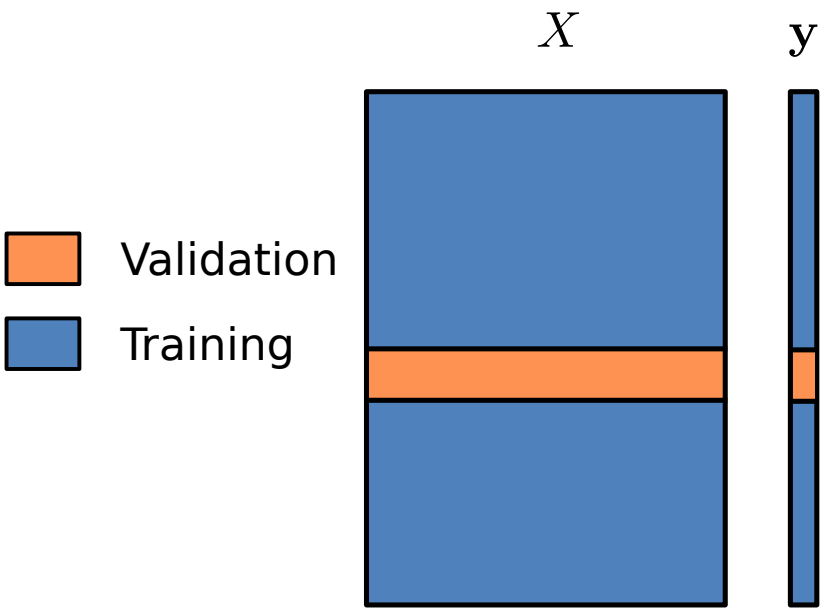


$$k = 5$$

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):

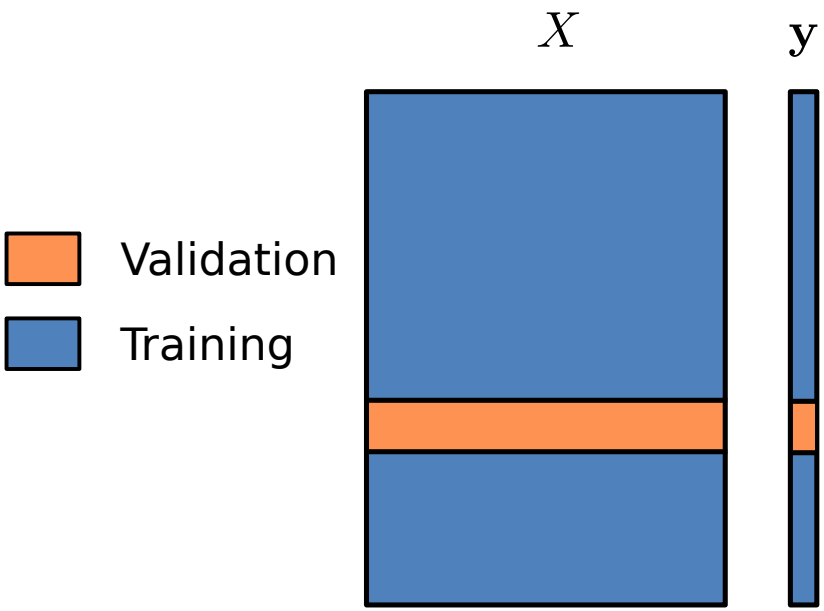


$$k = 6$$

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):

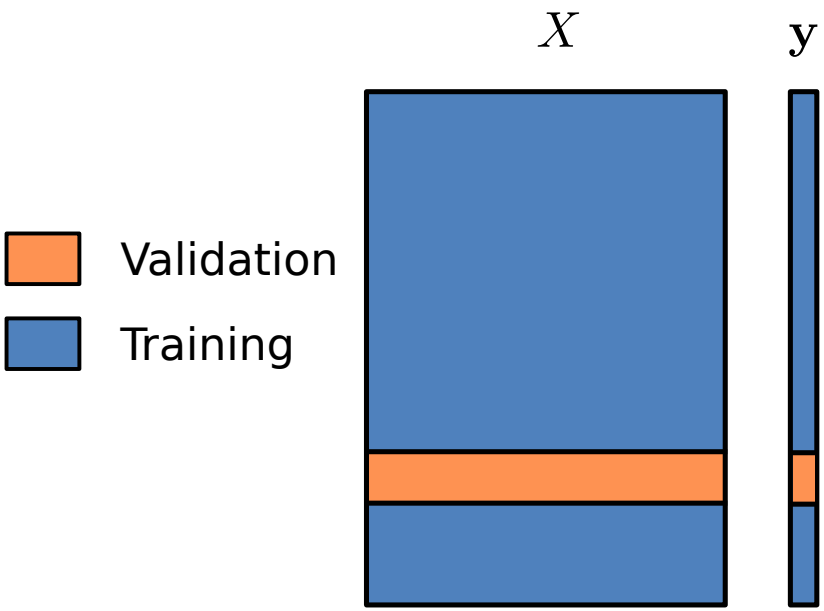


$$k = 7$$

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):

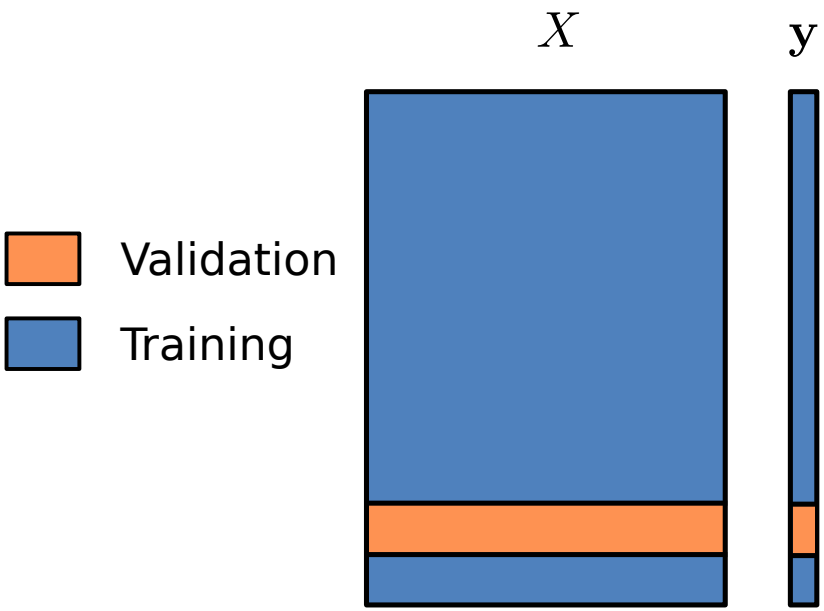


$$k = 8$$

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):

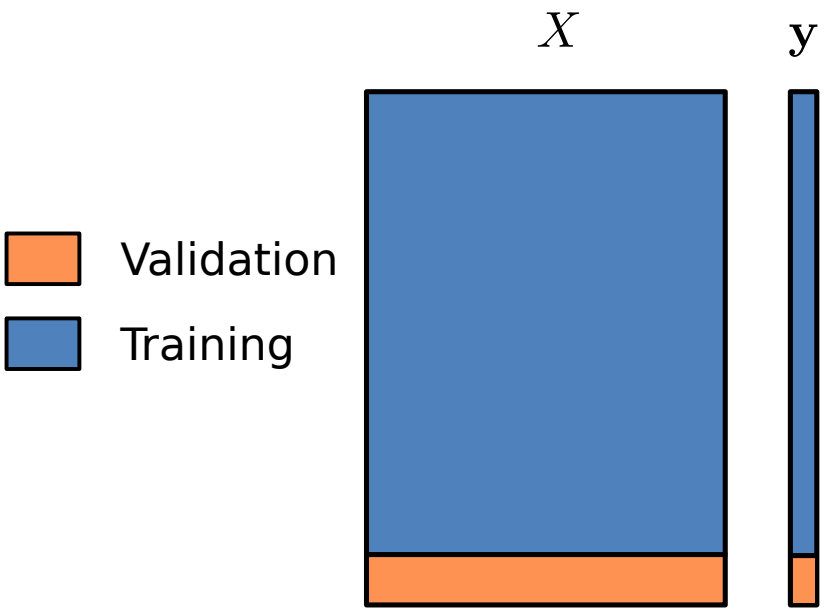


$$k = 9$$

1. Compute $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ 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):

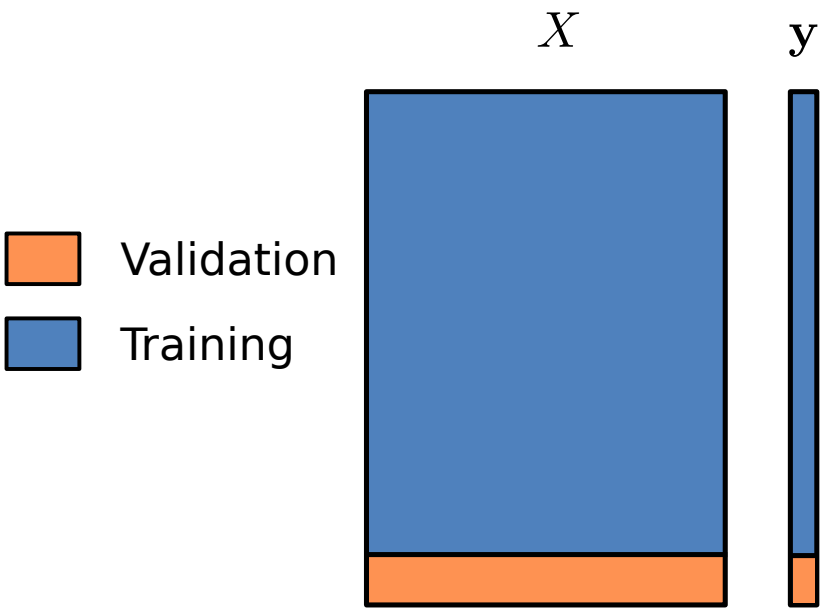


$$k = 10$$

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



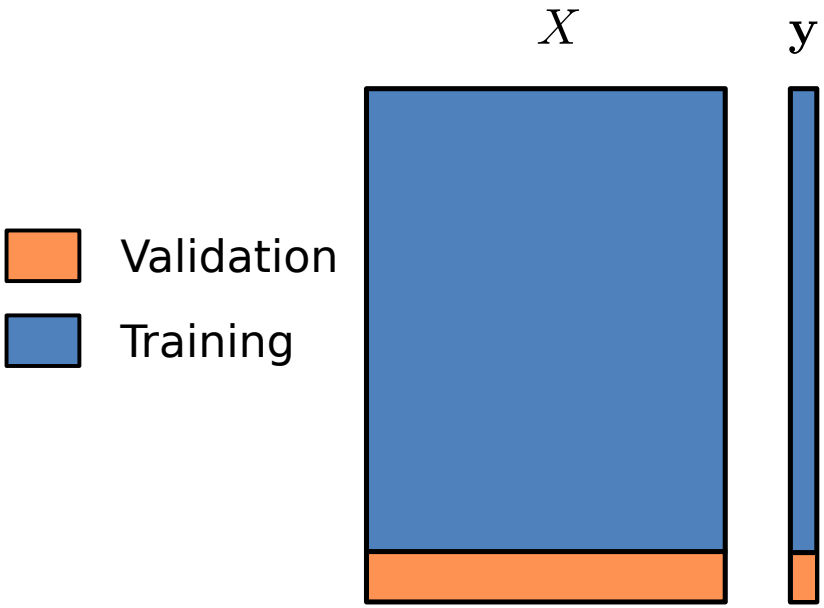
$k = 10$

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

Parameter choice: compute $\widehat{\text{Error}}_1, \ldots, \widehat{\text{Error}}_T$, average the errors and choose $\hat{i}^{\text{CV}} \in [\![1, T]\!]$ 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 $\hat{\beta}^{(\lambda_1)}, \ldots, \hat{\beta}^{(\lambda_T)}$ on train
2. Evaluate $\text{Error}_1^k, \ldots, \text{Error}_T^k$ on validation

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

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

Hyperparameter optimization (HO)

Possible selection criterion:

- Cross-validation[4] and its many variants
- AIC[5]
- BIC[6]
- SURE[7]
- etc.

---

[4] L. R. A. Stone and J.C. Ramer. "Estimating WAIS IQ from Shipley Scale scores: Another cross-validation". In: Journal of clinical psychology 21.3 (1965), pp. 297–297.

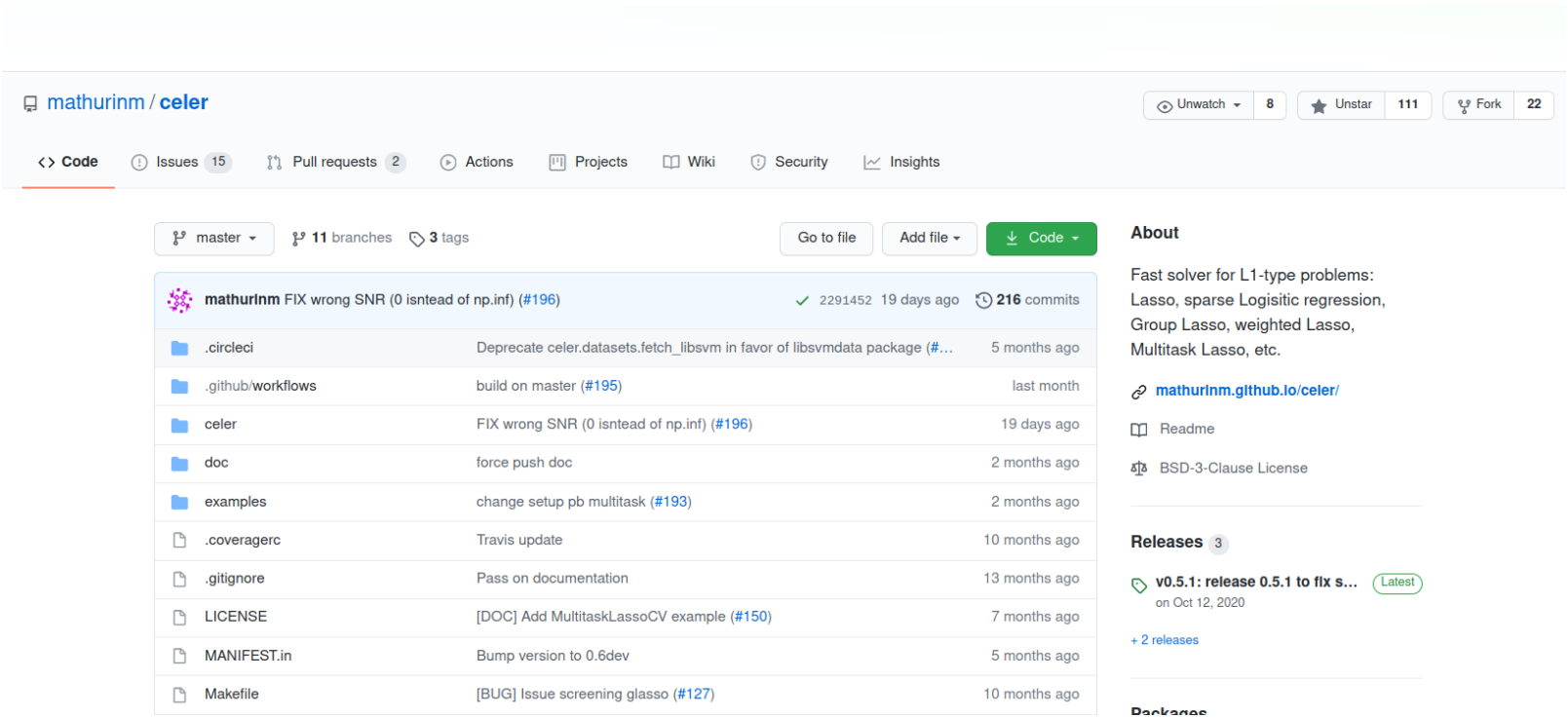
[5] H. Akaike. "A new look at the statistical model identification". In: IEEE Trans. Automat. Control AC-19 (1974), pp. 716–723.

[6] G. Schwarz. "Estimating the dimension of a model". In: Ann. Statist. 6.2 (1978), pp. 461–464.

[7] C. M. Stein. "Estimation of the mean of a multivariate normal distribution". In: Ann. Statist. 9.6 (1981), pp. 1135–1151.

Cross-validation / time

CV: time consuming, even for fast solvers
https://github.com/mathurinm/celer[8]:



[8] In 2020, Mathurin Massias got a PGMO Thesis Award + 2nd prize for Télécom Paris PhD Thesis Awards for it

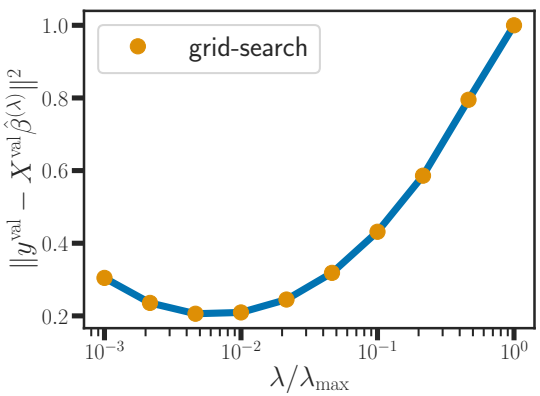
Hyperparameter optimization (HO)

For simplicity, let us focus on hold-out validation[9]:
Split $X, \mathbf{y}$ into $X^{\text{train}}, \mathbf{y}^{\text{train}}$ and $X^{\text{test}}, \mathbf{y}^{\text{test}}$

[9] cross-validation works similarly, but is less notationally friendly, see Appendix

Hyperparameter optimization (HO)

For simplicity, let us focus on hold-out validation[9]:
Split $X, \mathbf{y}$ 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

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

[9] cross-validation works similarly, but is less notationally friendly, see Appendix

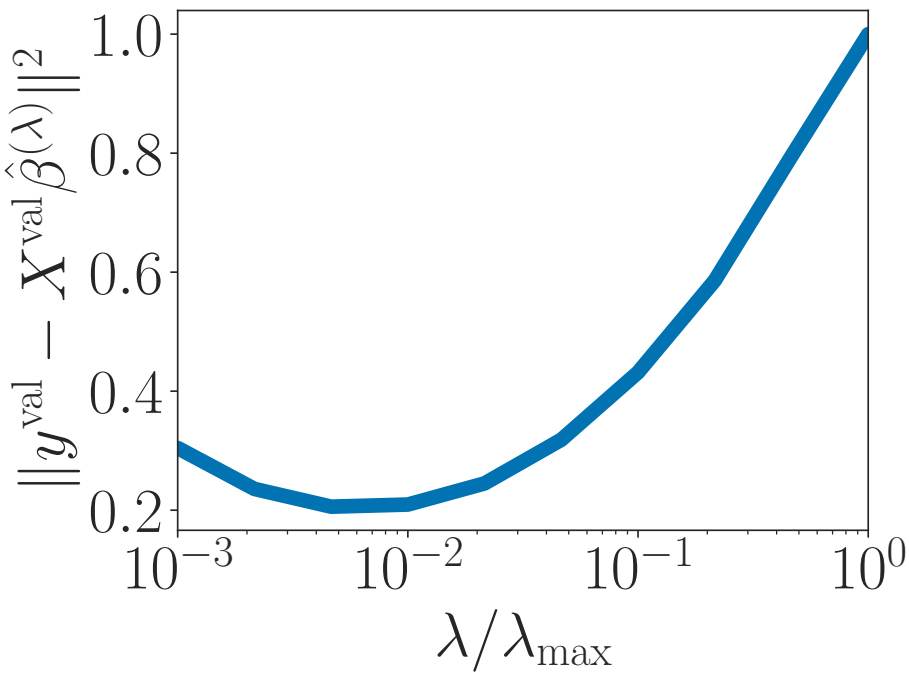
HO as a bilevel optimization problem[10][11]

outer optimization problem

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

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

inner optimization problem



Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

[10] 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.

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

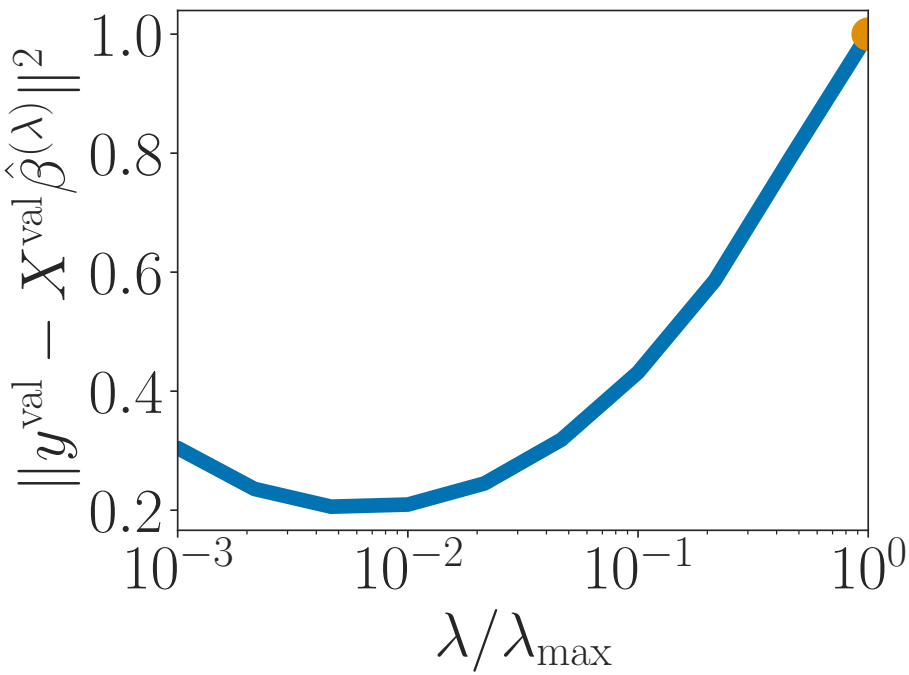
HO as a bilevel optimization problem[10][11]

outer optimization problem

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

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

inner optimization problem



Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

[10] 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.

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

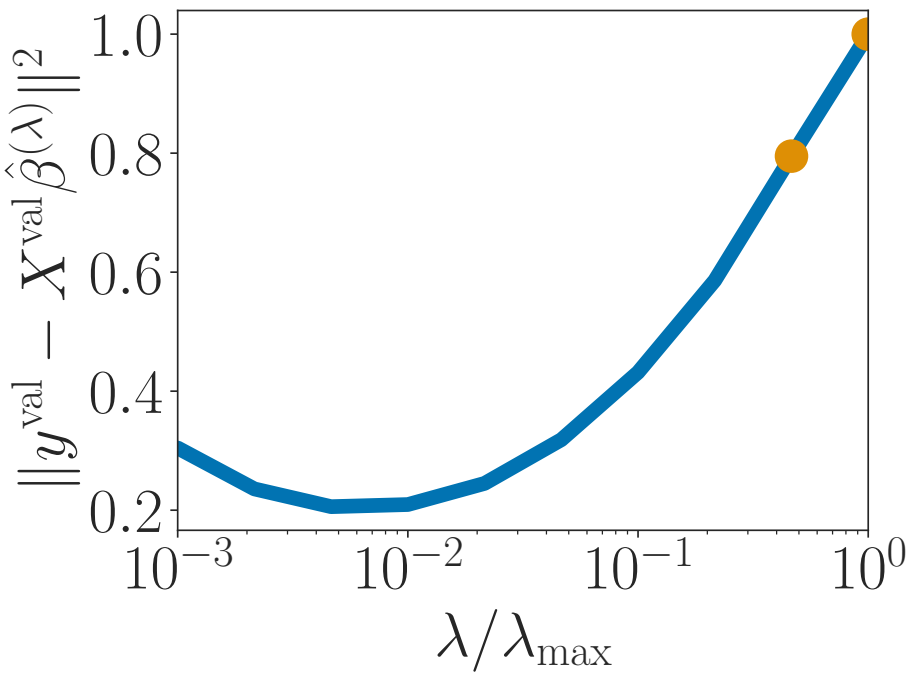
HO as a bilevel optimization problem[10][11]

outer optimization problem

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

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

inner optimization problem



Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

[10] 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.

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

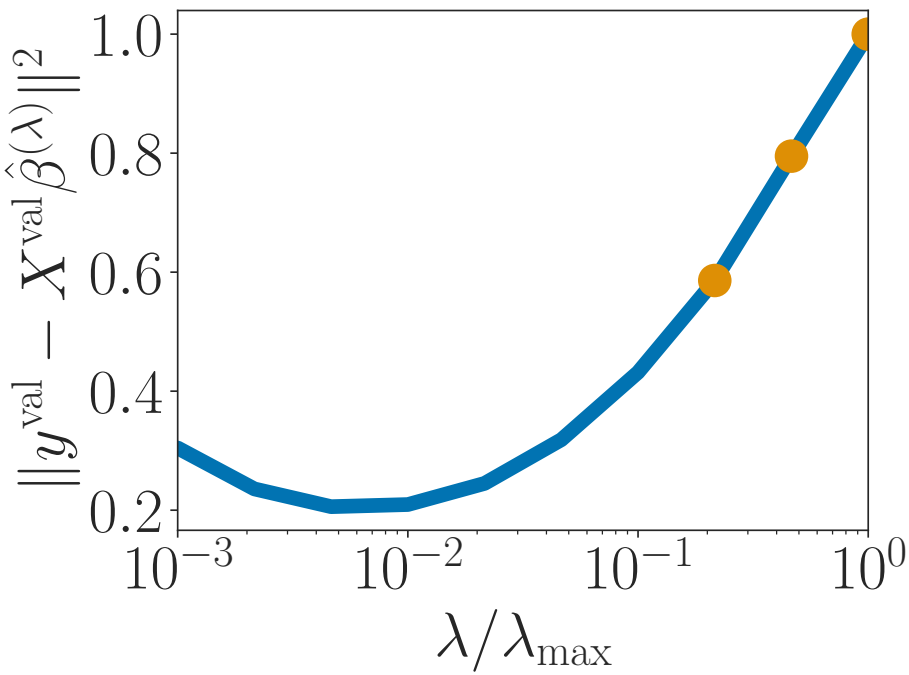
HO as a bilevel optimization problem[10][11]

outer optimization problem

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

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

inner optimization problem



Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

[10] 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.

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

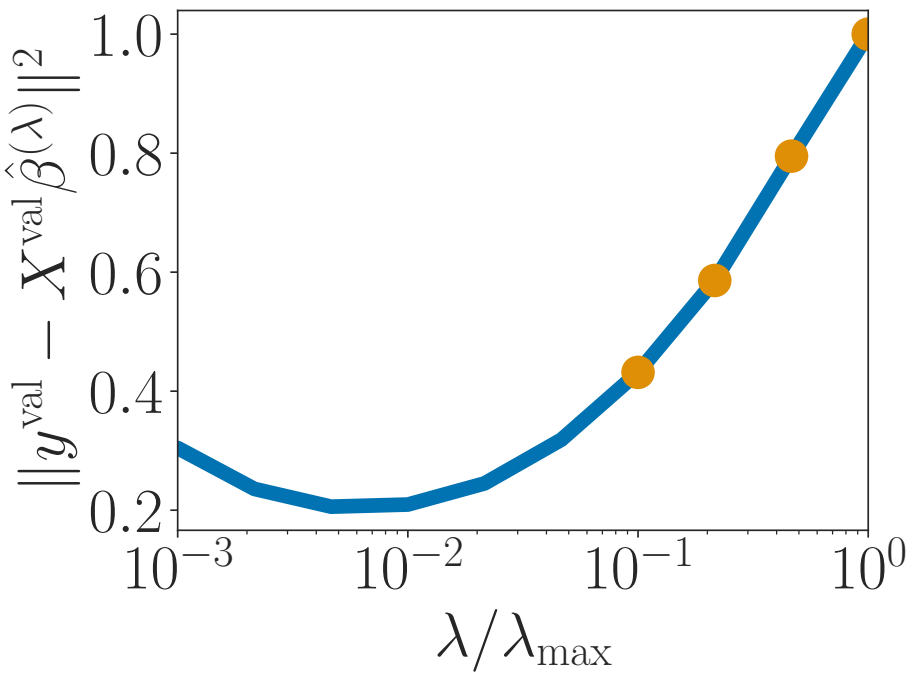
HO as a bilevel optimization problem[10][11]

outer optimization problem

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

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

inner optimization problem



Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

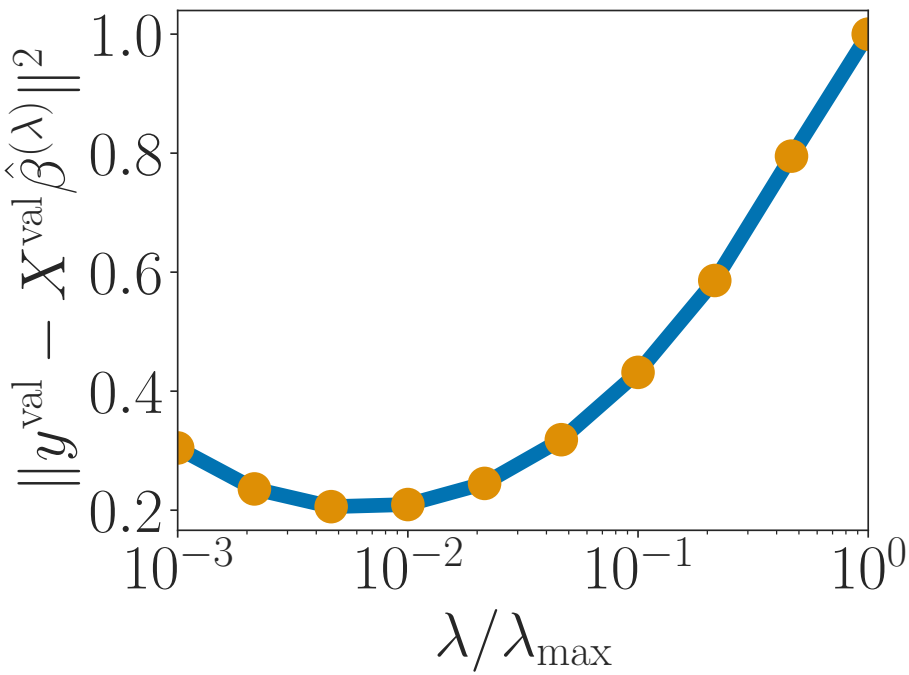
[10] P. Ochs et al. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: SIAM Journal on Imaging Sciences 8.1 (2015), pp. 331–372.

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

HO as a bilevel optimization problem[10][11]

$$\underbrace{\arg\min_{\lambda \in \mathbb{R}} \left\{ \mathcal{L}(\lambda) := \|y^{\text{val}} - X^{\text{val}} \hat{\beta}^{(\lambda)}\|^2 \right\}}_{\text{outer optimization problem}}$$

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

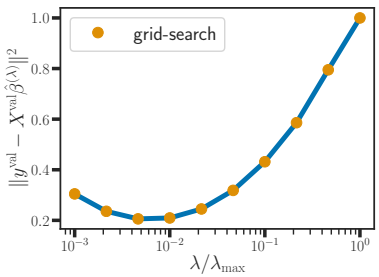


Grid choice : here default from `sklearn` or `glmnet` (it's an art...)

[10] 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.

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

Grid-search as a 0-order optimization method



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

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

- Grid-search, random-search,[12] SMBO[13]:
  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[14]:
  $\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla_\lambda \mathcal{L}(\lambda^{(t)})$ with suitable $\rho > 0$

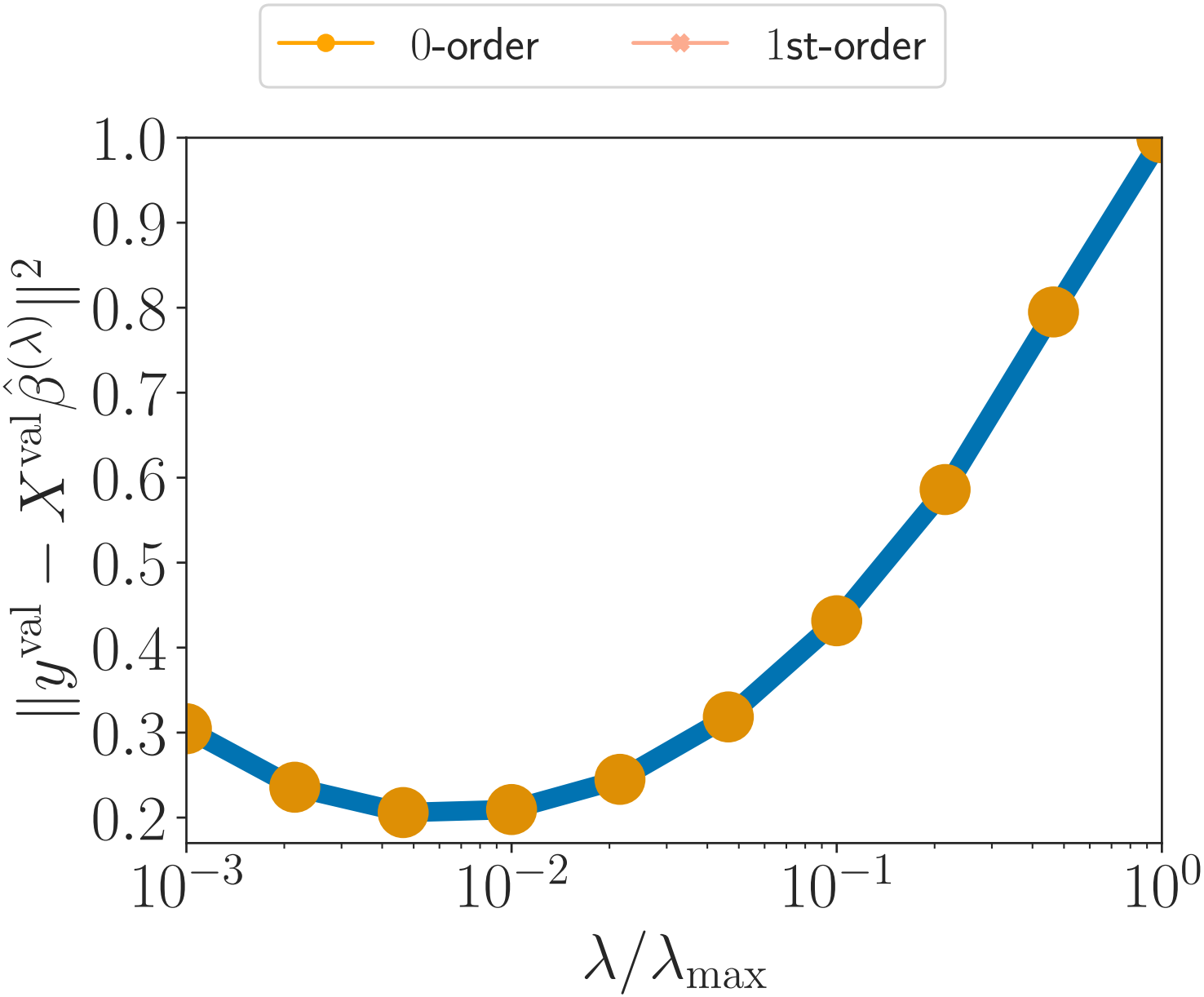
---

[12] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". In: Journal of Machine Learning Research 13.2 (2012).

[13] 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).

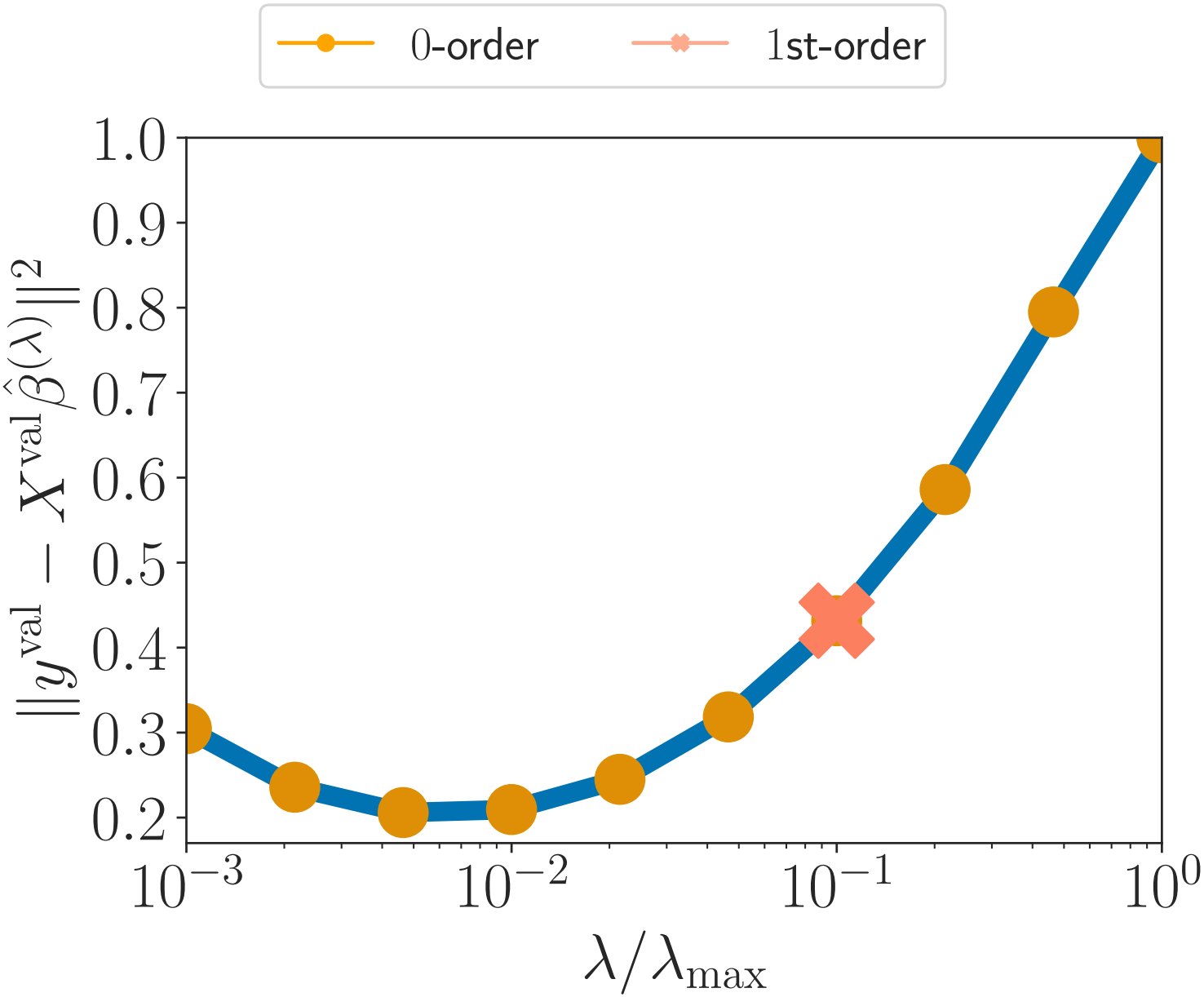
[14] F. Pedregosa. "Hyperparameter optimization with approximate gradient". In: ICML. vol. 48. 2016, pp. 737–746.

First-order optimization in λ, Lasso



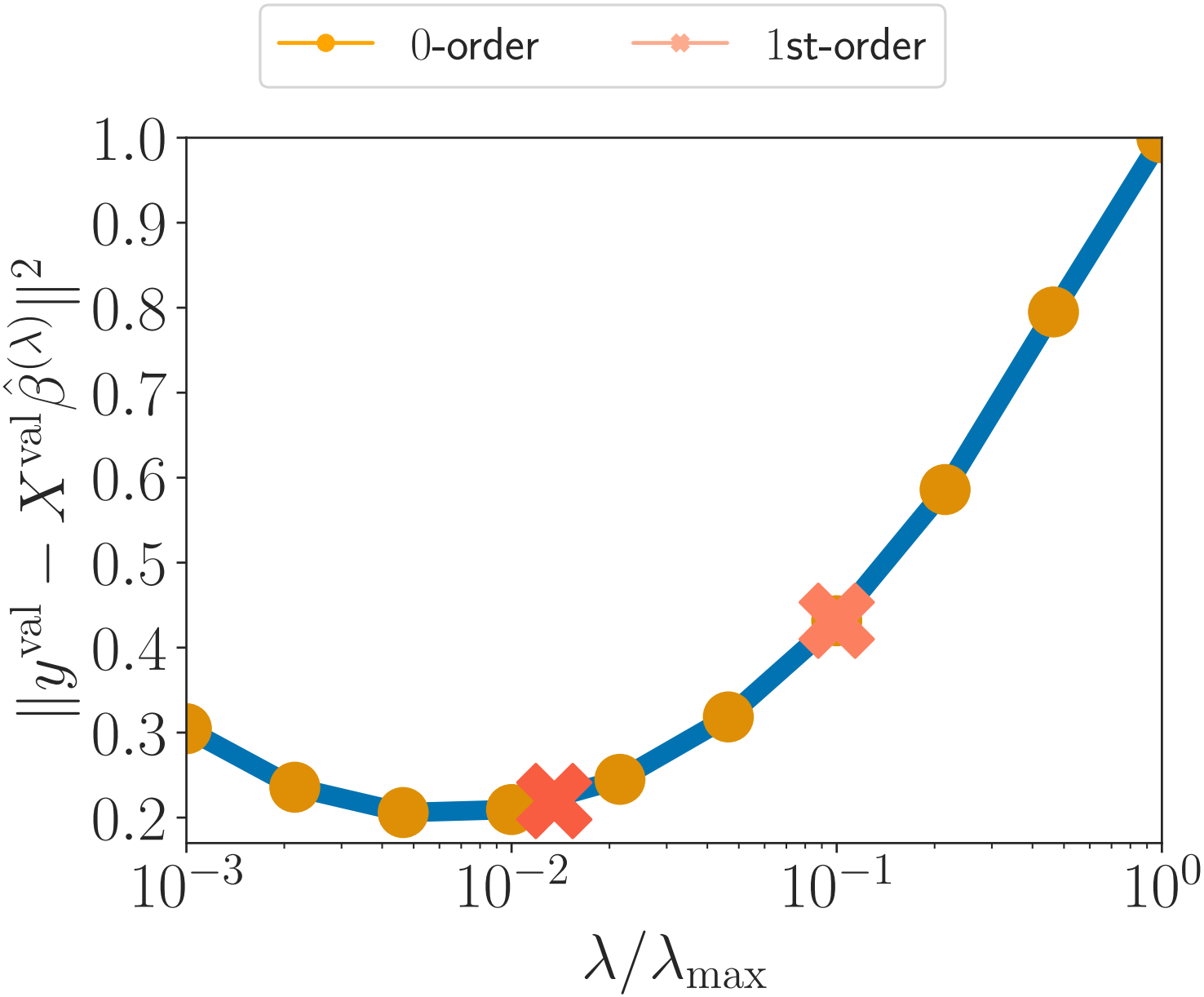
Real-sim dataset. Validation loss as a function of λ.

First-order optimization in λ, Lasso



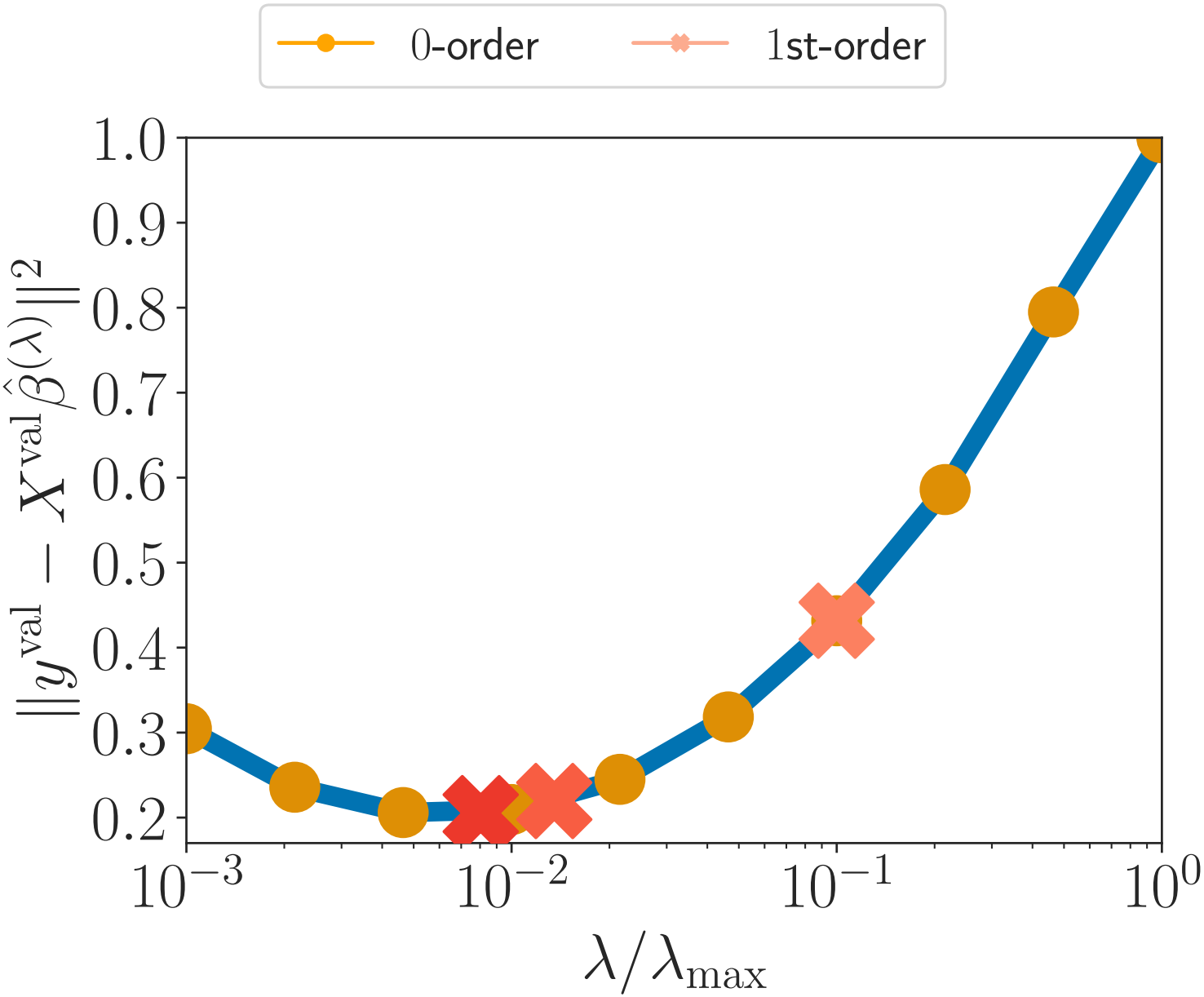
Real-sim dataset. Validation loss as a function of λ.

First-order optimization in λ, Lasso



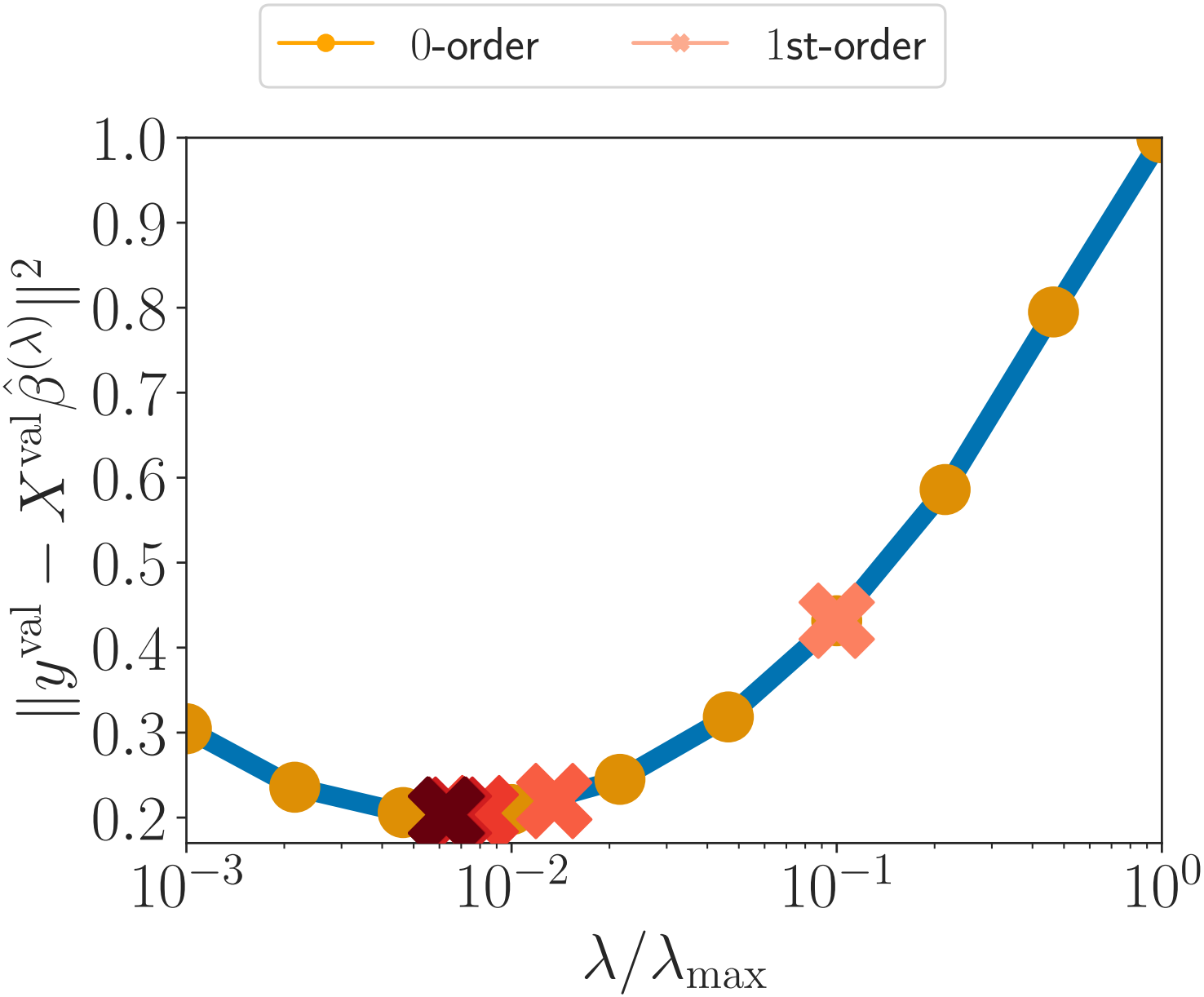
Real-sim dataset. Validation loss as a function of λ.

First-order optimization in λ, Lasso



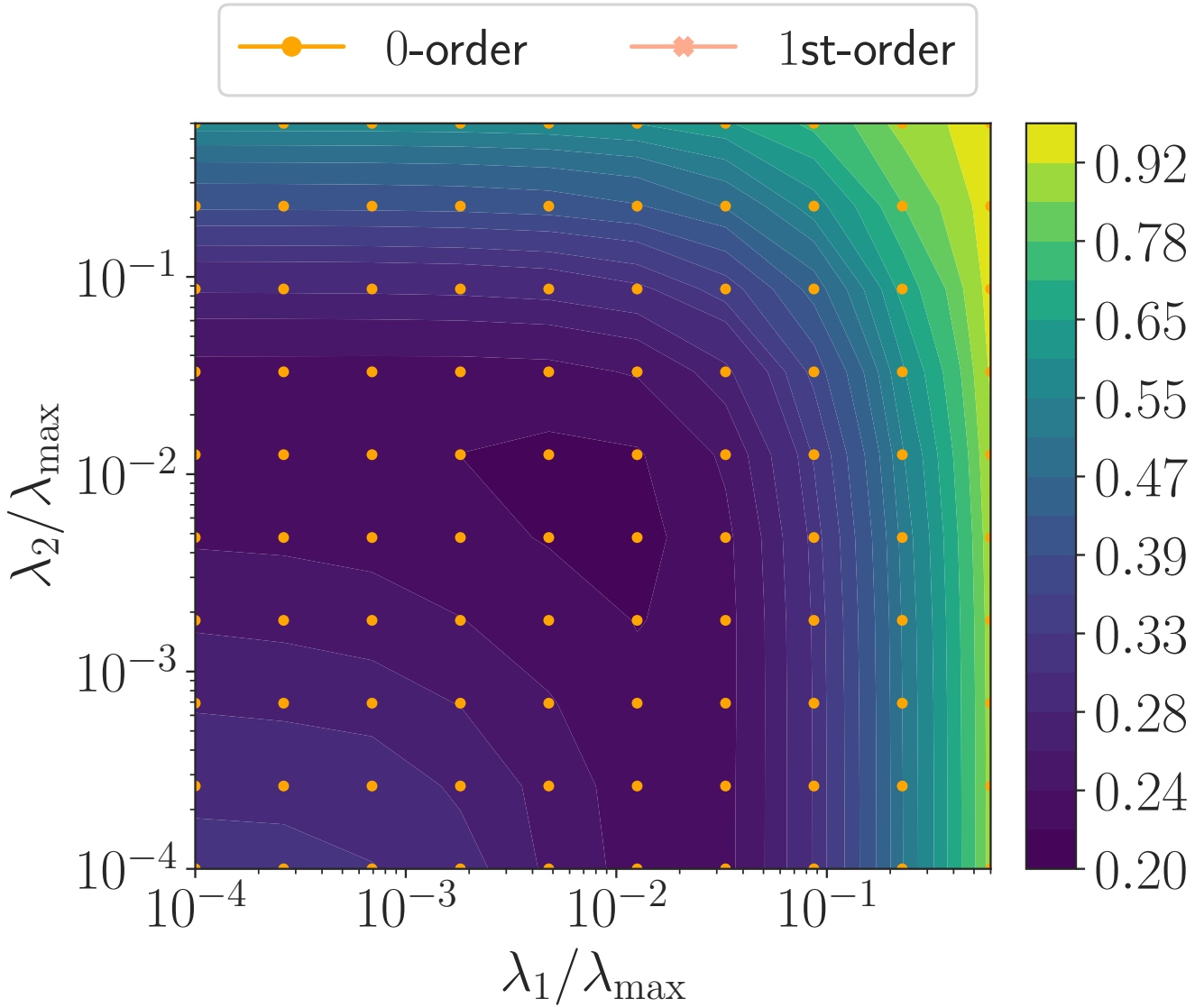
Real-sim dataset. Validation loss as a function of λ.

First-order optimization in λ, Lasso



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

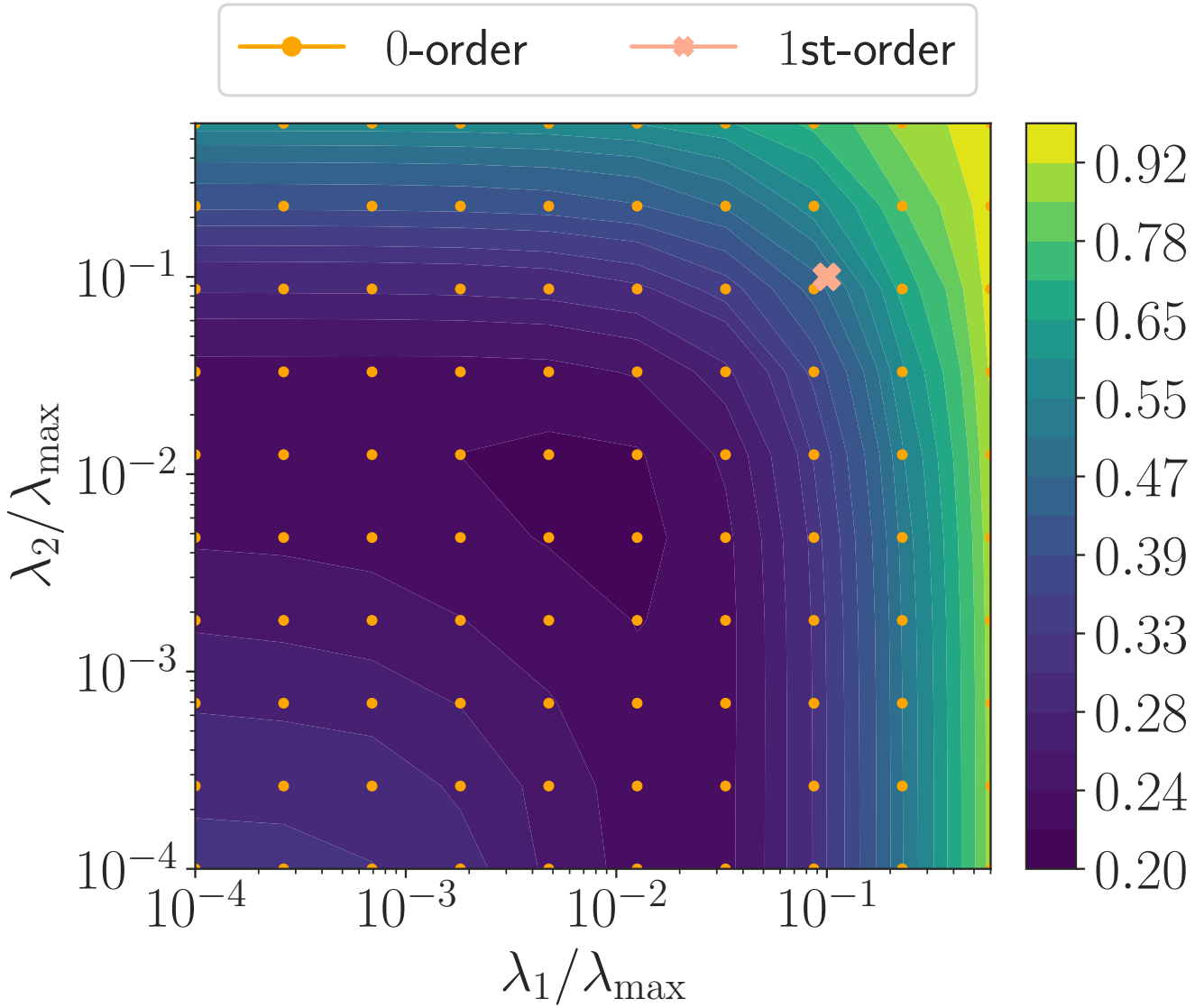
Who cares about 1D, what about Enet?



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

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

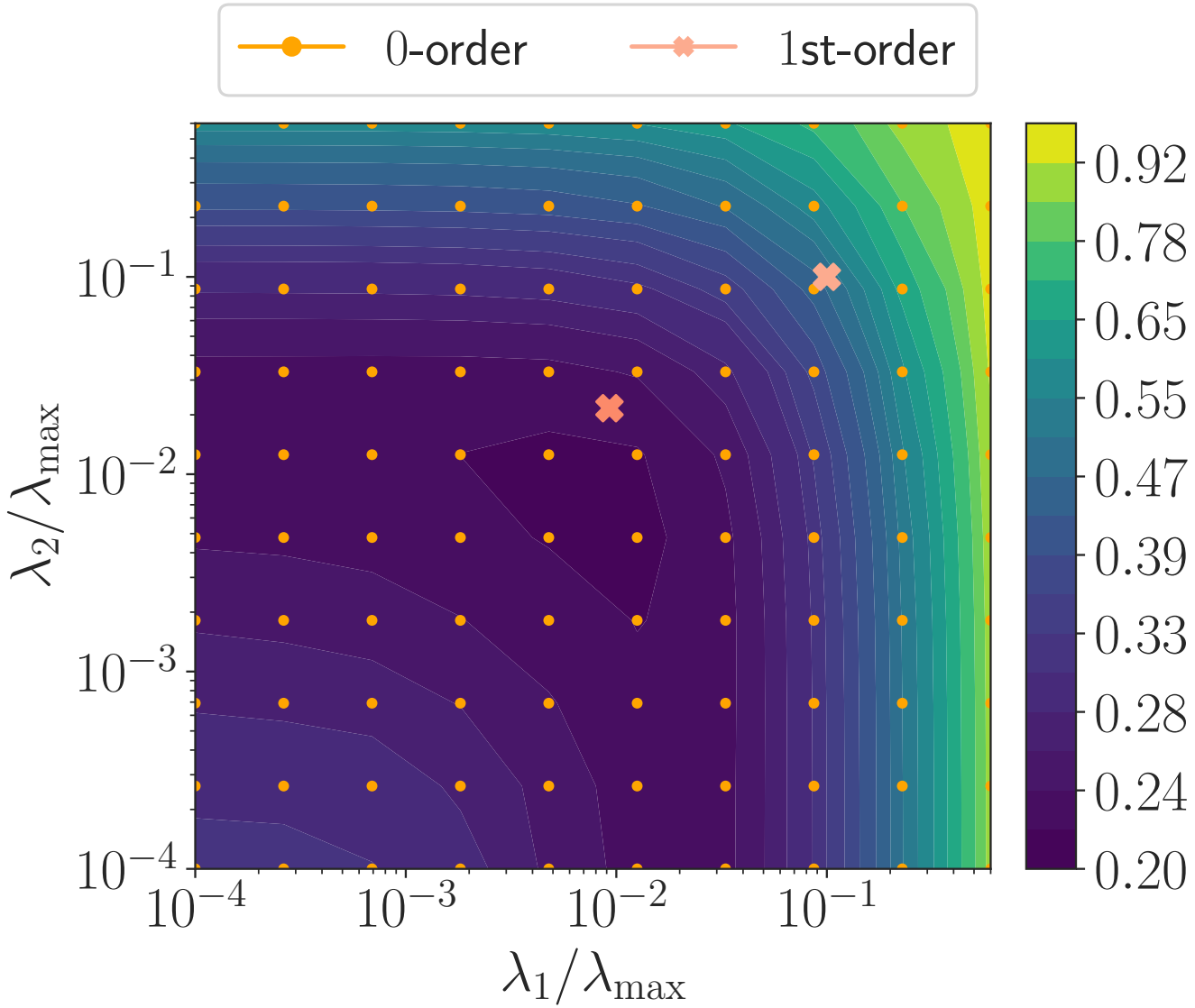
Who cares about 1D, what about Enet?



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

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

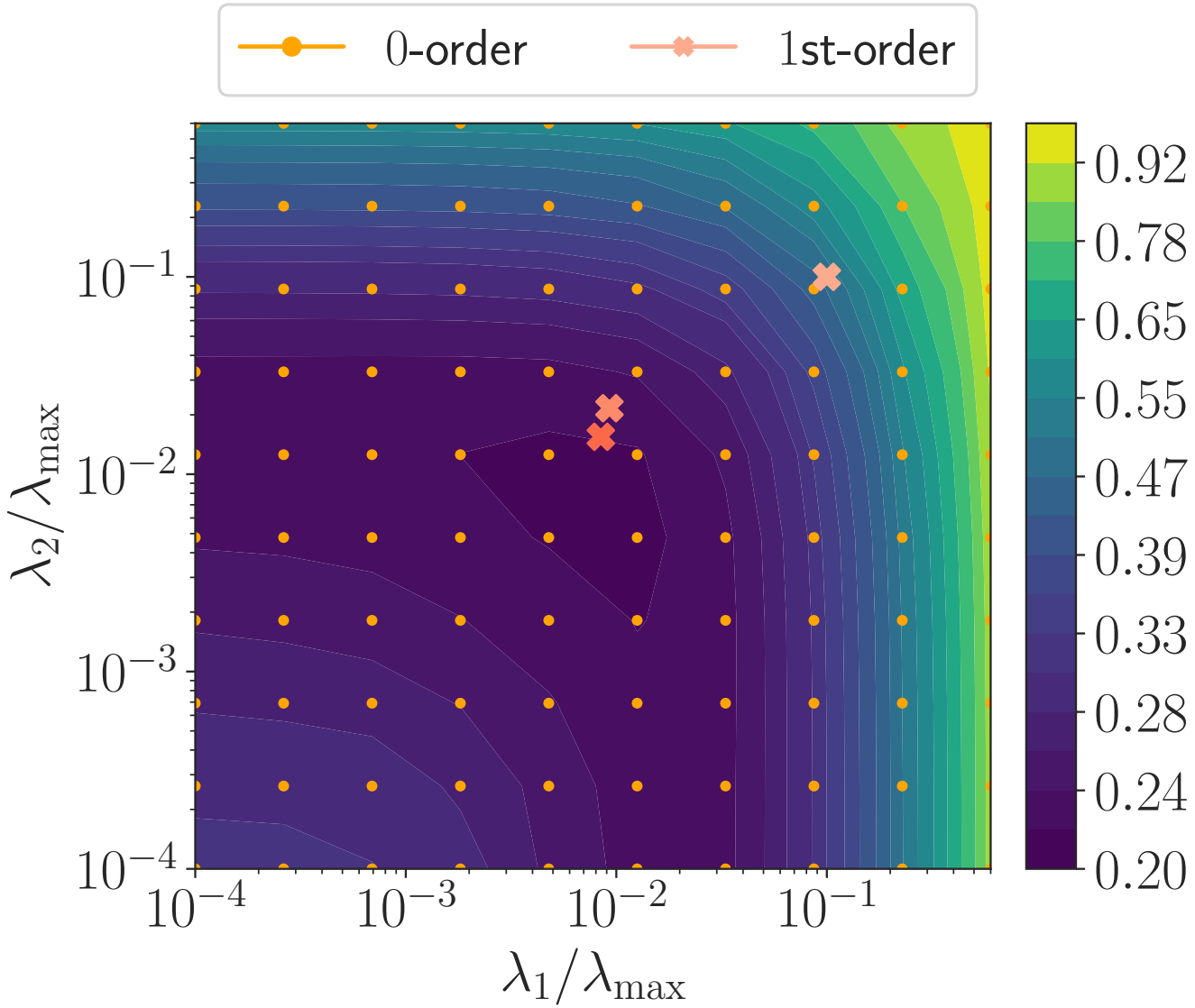
Who cares about 1D, what about Enet?



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

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

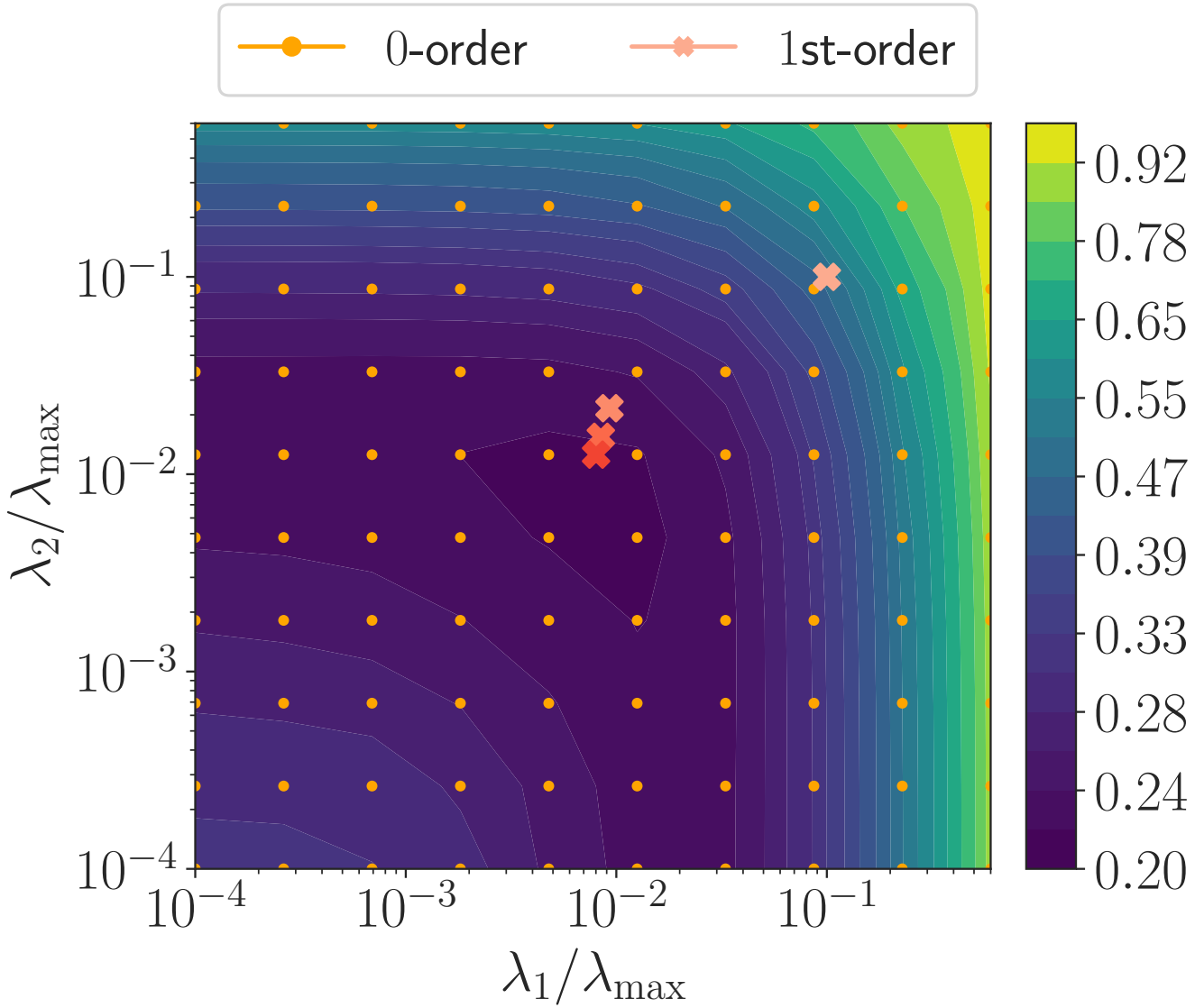
Who cares about 1D, what about Enet?



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

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

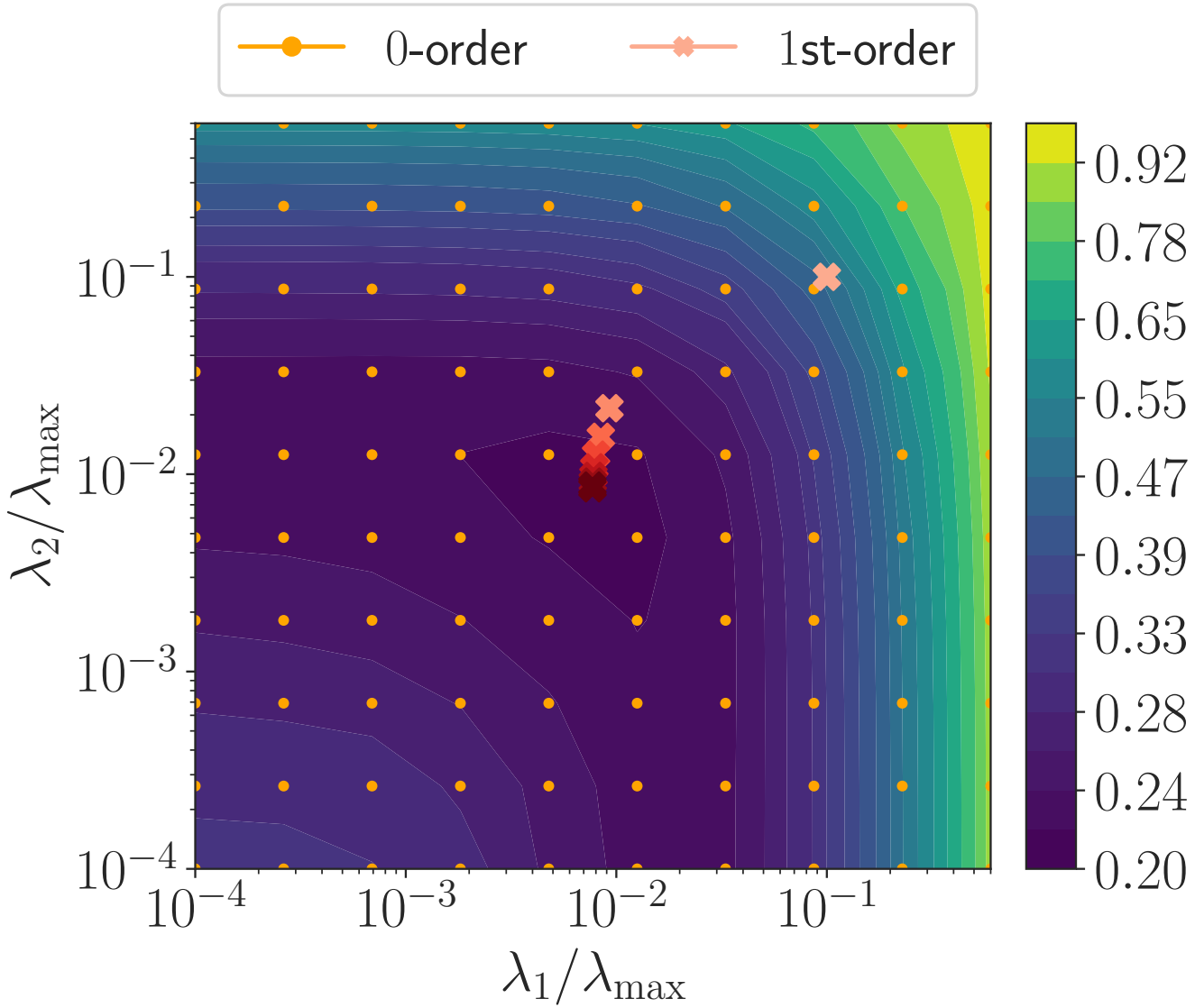
Who cares about 1D, what about Enet?



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

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

Who cares about 1D, what about Enet?



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

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

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

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

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

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

- Gradient descent
- possibly with line-search[15]
- LBFGS[16]
- etc.

---

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

[16] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: Mathematical programming 45.1-3 (1989), pp. 503–528.

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

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

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

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

- Gradient descent
- possibly with line-search[15]
- LBFGS[16]
- etc.

Main challenge: compute $\nabla_\lambda \mathcal{L}(\lambda)$ for a given λ

---

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

[16] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: Mathematical programming 45.1-3 (1989), pp. 503–528.

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

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

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

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

- Gradient descent
- possibly with line-search[15]
- LBFGS[16]
- etc.

Main challenge: compute $\nabla_\lambda \mathcal{L}(\lambda)$ for a given λ

---

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

[16] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: Mathematical programming 45.1-3 (1989), pp. 503–528.

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

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

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

Chain rule and Jacobian:

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

$\rightarrow$main challenge

- Boils down to:

  how to compute the Jacobian $\hat{\mathcal{J}}_{(\lambda)}$ efficiently?

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

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

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

Chain rule and Jacobian:

$$\nabla_\lambda \mathcal{L}(\lambda) = \underbrace{\hat{\mathcal{J}}_{(\lambda)}^\top}_{:=(\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \ldots, \nabla_\lambda \hat{\beta}_p^{(\lambda)}) \\ \rightarrow \text{main challenge}} \nabla_\beta C(\hat{\beta}^{(\lambda)})$$

- Boils down to:

  how to compute the Jacobian $\hat{\mathcal{J}}_{(\lambda)}$ efficiently?

How to compute $\hat{\mathcal{J}}_{(\lambda)} := (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top$?

$$\underbrace{\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y^{\text{train}} - X^{\text{train}} \beta\|^2 + \frac{\lambda}{2} \|\beta\|^2}_{\text{inner optimization problem}}$$

"Smooth" inner optimization problems, **well studied**:

- Implicit differentiation (**closed-form** formula)[17]:
  need to solve a $p \times p$ linear system ($p = \#\text{features}$)
  details in Appendix
- Automatic differentiation, forward[18] or reverse[19] mode

---

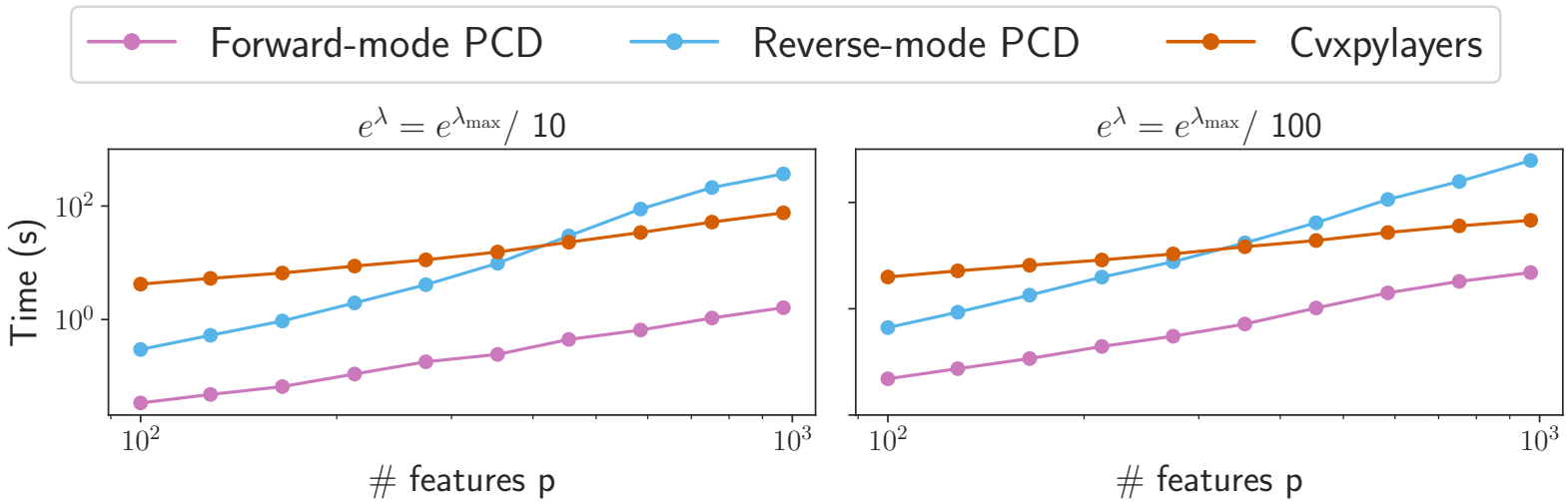
[17] J. Larsen et al. "Design and regularization of neural networks: the optimal use of a validation set". In: Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop. 1996; Y. Bengio. "Gradient-based optimization of hyperparameters". In: Neural computation 12.8 (2000), pp. 1889–1900.

[18] L. Franceschi et al. "Forward and reverse gradient-based hyperparameter optimization". In: ICML. 2017, pp. 1165–1173.

[19] J. Domke. "Generic methods for optimization-based modeling". In: AISTATS. vol. 22. 2012, pp. 318–326.

Automatic differentiation out of the box?

Automatic differentiation e.g., `cvxpylayers`[20] a software based on `cvxpy`,[21] solving disciplined parametrized programming and providing derivatives w.r.t. the parameters of the program



Lasso hold-out, time to compute $\nabla_\lambda \mathcal{L}(\lambda)$, gina dataset. Time to compute a single hypergradient as a function of the number of features

Conclusion: too slow for our high dimensional problems

---

[20] A. Agrawal et al. "Differentiable convex optimization layers". In: NeurIPS. 2019, pp. 9558–9570.

[21] S. Diamond and S. Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: Journal of Machine Learning Research 17.83 (2016), pp. 1–5.

How to compute $\hat{\mathcal{J}}_{(\lambda)} := (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top$?

$$\underbrace{\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1}_{\text{inner optimization problem}}$$

"Nonsmooth" inner optimization problems, scarce literature:

- Smooth the nonsmooth term[22]
- Use algorithms with differentiable updates[23][24] (Bregman)

Our contributions:

- Iterative differentiation can be applied on proximal algorithms
- $\hat{\mathcal{J}}_{(\lambda)} := (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top$ shares $\hat{\beta}^{(\lambda)}$'s **sparsity pattern**

[22] G. Peyré and J. M. Fadili. "Learning analysis sparsity priors". In: Sampta. 2011.

[23] P. Ochs et al. "Bilevel optimization with nonsmooth lower level problems". In: SSVM. 2015, pp. 654–665.

[24] J. Frecon, S. Salzo, and M. Pontil. "Bilevel learning of the group lasso structure". In: NeurIPS. 2018, pp. 8301–8311.

How to compute $\hat{\mathcal{J}}_{(\lambda)} := (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top$?

$$\underbrace{\hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y^{\text{train}} - X^{\text{train}} \beta\|^2 + \lambda \|\beta\|_1}_{\text{inner optimization problem}}$$

"Nonsmooth" inner optimization problems, scarce literature:

- Smooth the nonsmooth term[22]
- Use algorithms with differentiable updates[23][24] (Bregman)

Our contributions:

- Iterative differentiation can be applied on proximal algorithms
- $\hat{\mathcal{J}}_{(\lambda)} := (\nabla_\lambda \hat{\beta}_1^{(\lambda)}, \dots, \nabla_\lambda \hat{\beta}_p^{(\lambda)})^\top$ shares $\hat{\beta}^{(\lambda)}$'s **sparsity pattern**

---

[22] G. Peyré and J. M. Fadili. "Learning analysis sparsity priors". In: Sampta. 2011.

[23] P. Ochs et al. "Bilevel optimization with nonsmooth lower level problems". In: SSVM. 2015, pp. 654–665.

[24] J. Frecon, S. Salzo, and M. Pontil. "Bilevel learning of the group lasso structure". In: NeurIPS. 2018, pp. 8301–8311.

Forward-mode differentiation of PGD

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \overbrace{f}^{\text{smooth}}(\beta) + \lambda \overbrace{g}^{\text{non-smooth}}(\beta) \tag{1}$$

Algorithm: Proximal gradient descent PGD

init : $\beta = 0_p,$, L

for $\underline{\text{iter} = 1, \ldots,}$ **do**

$z \leftarrow \beta - \frac{1}{L} \nabla f(\beta)$ // gradient step

$\beta \leftarrow \text{prox}_{\lambda g / L}(z)$ // proximal step

return $\underline{\beta}$

Forward-mode differentiation of PGD

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \overbrace{f}^{\text{smooth}}(\beta) + \lambda \overbrace{g}^{\text{non-smooth}}(\beta) \tag{1}$$

Algorithm: Forward-mode differentiation of PGD

init : $\beta = 0_p,$, L

for $\underline{\text{iter} = 1, \ldots,}$ **do**

$z \leftarrow \beta - \frac{1}{L} \nabla f(\beta)$ // gradient step

$dz \leftarrow \left(\mathrm{Id}_p - \frac{1}{L} \nabla^2 f(\beta) \right) \mathcal{J}$ // diff w.r.t. λ: chain rule

$\beta \leftarrow \mathrm{prox}_{\lambda g / L}(z)$ // proximal step

return $\underline{\beta}$

Forward-mode differentiation of PGD

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \overbrace{f}^{\text{smooth}}(\beta) + \lambda \overbrace{g}^{\text{non-smooth}}(\beta) \tag{1}$$

Algorithm: Forward-mode differentiation of PGD

init : $\beta = 0_p$, $\mathcal{J} = 0_p$, L

for $\text{iter} = 1, \ldots,$ **do**

$z \leftarrow \beta - \frac{1}{L} \nabla f(\beta)$ // gradient step

$dz \leftarrow \left(\text{Id}_p - \frac{1}{L} \nabla^2 f(\beta) \right) \mathcal{J}$ // diff w.r.t. λ: chain rule

$\beta \leftarrow \text{prox}_{\lambda g / L}(z)$ // proximal step

$\mathcal{J} \leftarrow \partial_z \text{prox}_{\lambda g / L}(z) dz$ // diff w.r.t. λ: chain rule

return β, $\mathcal{J}$

Forward-mode differentiation of PGD

$$\hat{\beta}^{(\lambda)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \overbrace{f}^{\text{smooth}}(\beta) + \lambda \overbrace{g}^{\text{non-smooth}}(\beta) \tag{1}$$

Algorithm: Forward-mode differentiation of PGD

init : $\beta = 0_p$, $\mathcal{J} = 0_p$, L

for $\text{iter} = 1, \ldots,$ **do**

$z \leftarrow \beta - \frac{1}{L} \nabla f(\beta)$ // gradient step

$dz \leftarrow \left(\mathrm{Id}_p - \frac{1}{L} \nabla^2 f(\beta) \right) \mathcal{J}$ // diff w.r.t. λ: chain rule

$\beta \leftarrow \mathrm{prox}_{\lambda g / L}(z)$ // proximal step

$\mathcal{J} \leftarrow \partial_z \mathrm{prox}_{\lambda g / L}(z) dz$ // diff w.r.t. λ: chain rule

$\quad + \partial_\lambda \mathrm{prox}_{\lambda g / L}(z)$ // do not forget this term!

return β, $\mathcal{J}$

Forward-mode differentiation of PCD

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

- Forward-mode differentiation can be also be applied on **proximal coordinate descent** (PCD),
- **Convergence** of the Jacobian sequence $\mathcal{J}$?

Forward-mode differentiation of PCD

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

- Forward-mode differentiation can be also be applied on **proximal coordinate descent** (PCD),
- **Convergence** of the Jacobian sequence $\mathcal{J}$?

Contribution

- Proved Jacobian sequence convergence for PGD and PCD

Forward-mode differentiation of PCD

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

- Forward-mode differentiation can be also be applied on **proximal coordinate descent** (PCD),
- **Convergence** of the Jacobian sequence $\mathcal{J}$?

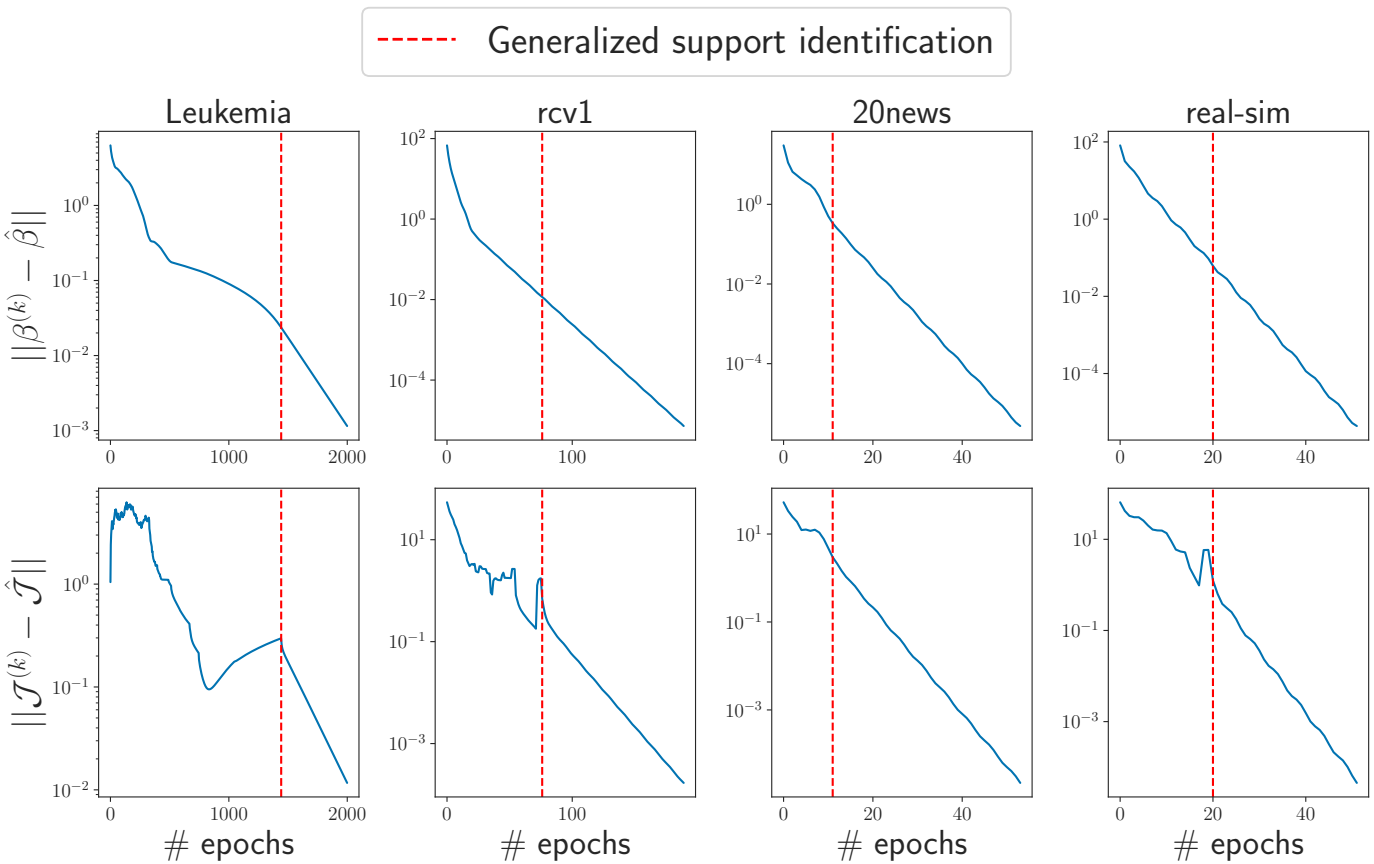
Contribution

- Proved Jacobian sequence convergence for PGD and PCD

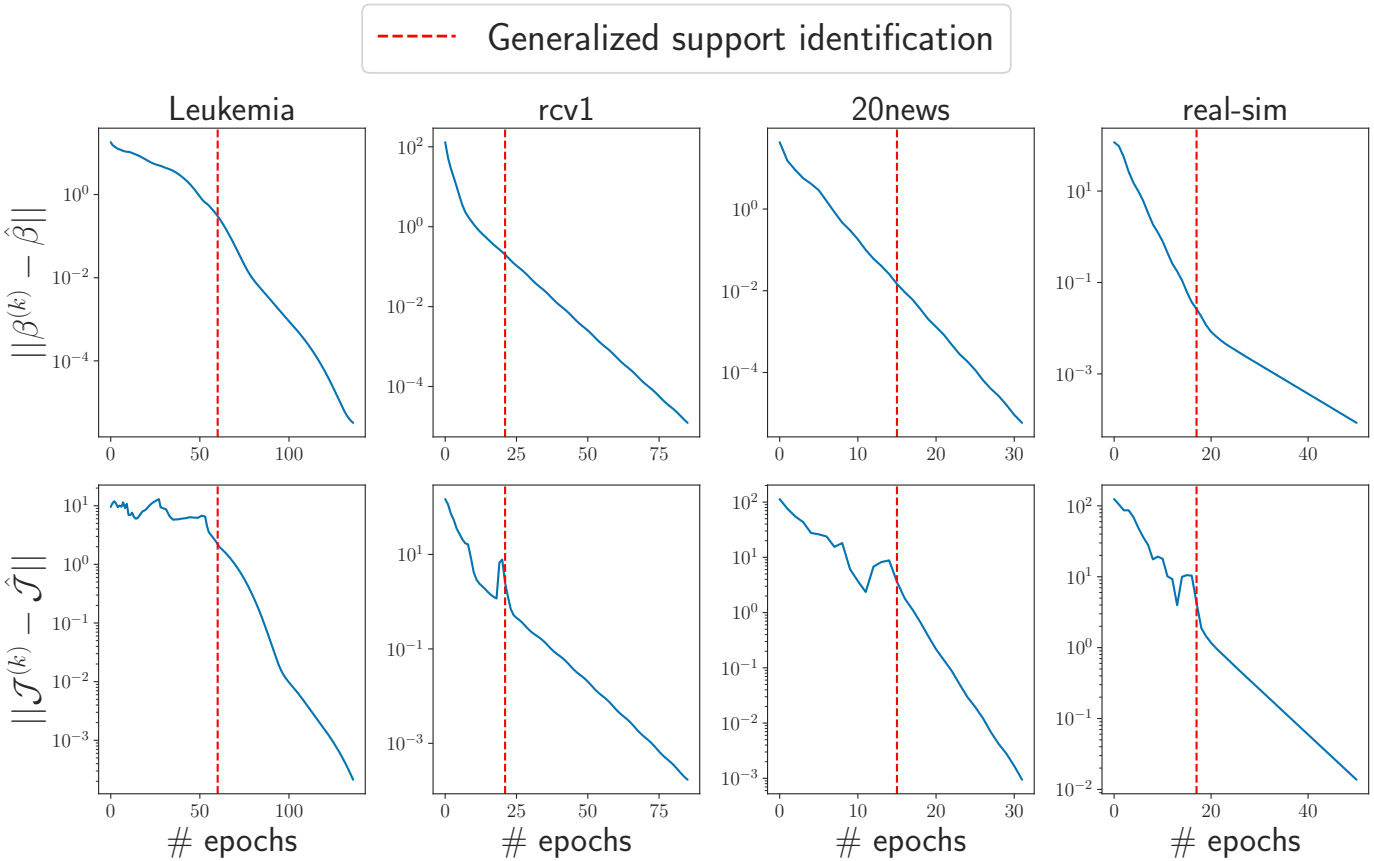
Local linear convergence of the Jacobian (I)

Proposition: forward diff. convergence (Lasso)

Assuming that the Lasso inner optimization has a unique minimizer, then the Jacobian sequence based on forward diff. of PCD converges to the true Jacobian. Once the support (i.e., non-zeros coefs.) has been identified, convergence is linear.



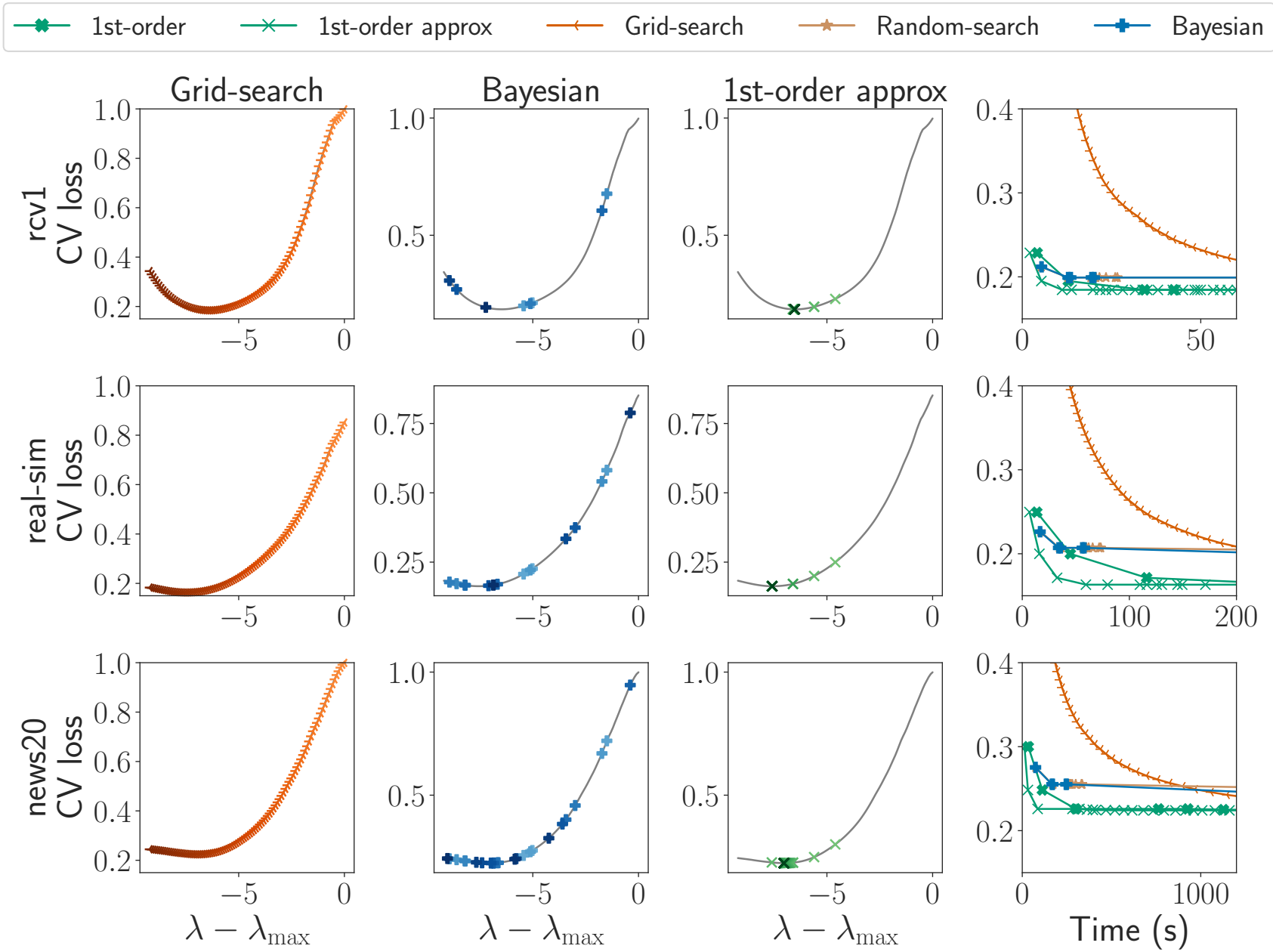
Local linear convergence of the Jacobian (II)



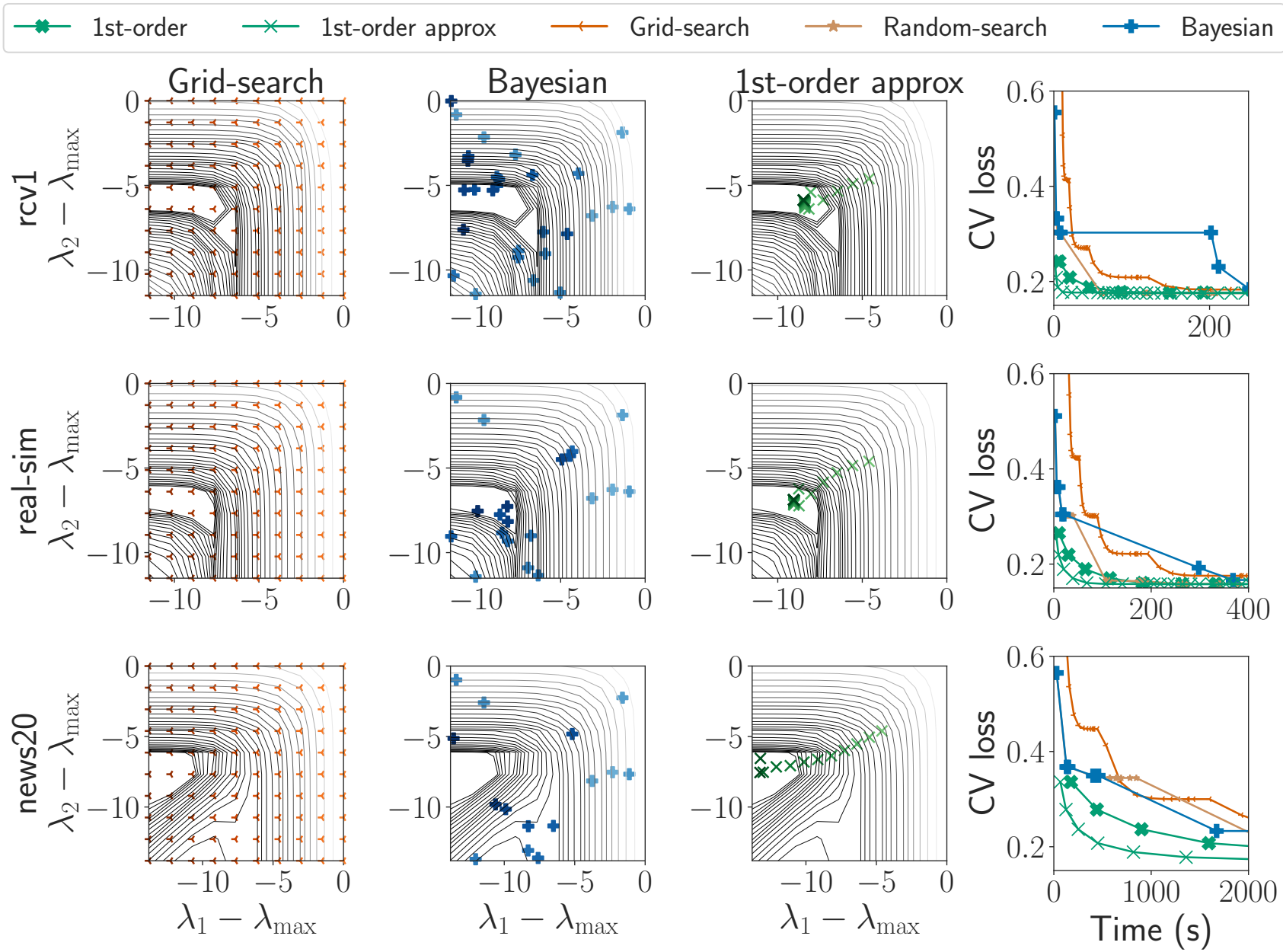
Example: sparse logistic regression

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_i \frac{1}{1 + \exp(-y_i X_{i:} \beta)} + \lambda \|\beta\|_1$$

Experiments I - Lasso cross-validation

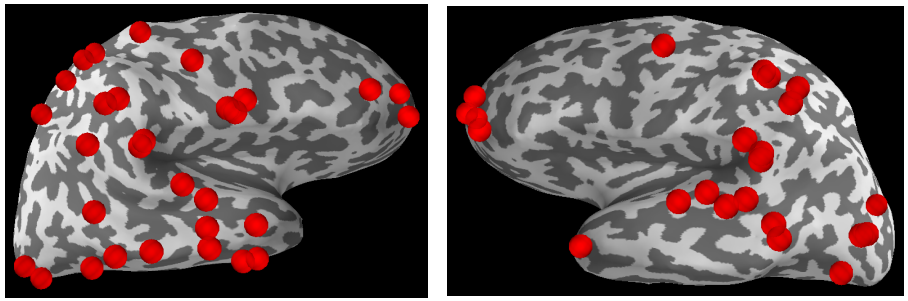


Experiments II - Enet cross-validation



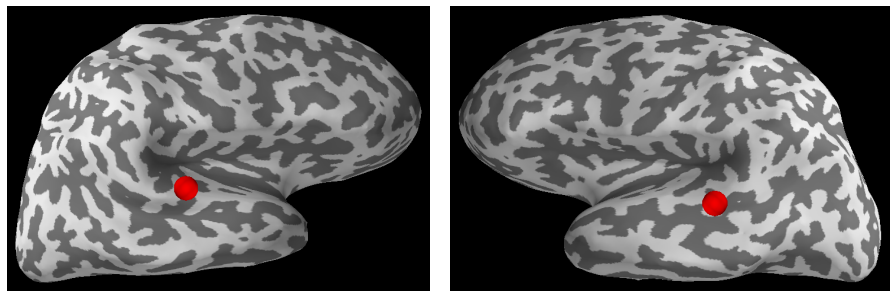
Experiments III - Real MEEG data

- **Outer criterion:** SURE
- **Inner problems:** the Lasso and weighted Lasso



Vanilla Lasso (1 parameter)

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} ||y - X\beta||_2^2 + \lambda \|\beta\|_1$$



Weighted Lasso (p parameters)

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} ||y - X\beta||_2^2 + \sum_{j=1}^{p} \lambda_j |\beta_j|$$

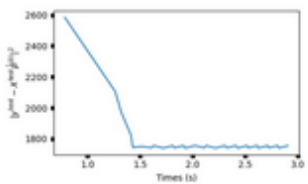
sparse-ho

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

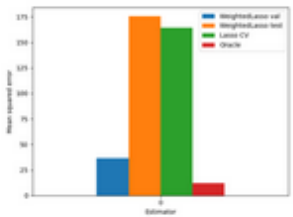
sparse-ho 0.1.dev Examples API GitHub Site

Examples Gallery

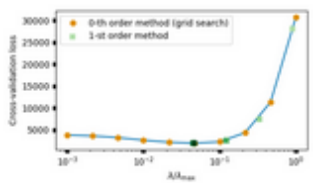
The examples gallery provides working code samples demonstrating what can be done with the sparse_ho library.



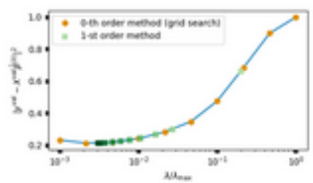
Monitor custom metrics along hyperoptimization



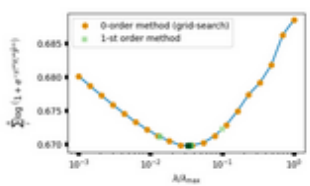
Weighted Lasso with held-out test set



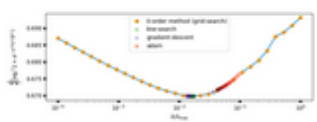
Lasso with Cross-validation



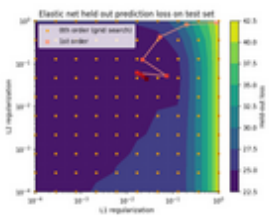
Lasso with held-out test set



Sparse logistic regression



Compare outer optimizers



Elastic net with held-out test set

Limitations

- Specific parametrization e^{λ} (avoid positivity constraints)
- Need a **differentiable criterion**: cannot use 0/1-loss
- Need a **continuous estimator** w.r.t. data and hyperparameters: does not apply yet to **non-convex** penalties[25] nor reweighted Lasso[26]
- Optimized function often **non-convex**: possibly multiple local minima

[25] P. Breheny and J. Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection". In: Ann. Appl. Stat. 5.1 (2011), p. 232.

[26] E. J. Candès, M. B. Wakin, and S. P. Boyd. "Enhancing Sparsity by Reweighted l_1 Minimization". In: J. Fourier Anal. Applicat. 14.5-6 (2008), pp. 877–905.

Conclusion

Contributions:

- 1st-order optimization with nonsmooth inner problem
- **Local linear convergence** of the Jacobian
- **Leverage sparsity** to speed up hypergradient computation

Future work:

- Convergence of the bilevel procedure
- Smarter outer solver

Conclusion

Contributions:

- 1st-order optimization with nonsmooth inner problem
- **Local linear convergence** of the Jacobian
- **Leverage sparsity** to speed up hypergradient computation

Future work:

- Convergence of the bilevel procedure
- Smarter outer solver

References:

- Long version on **arXiv** soon
- Open source package `https://github.com/QB3/sparse-ho`

Conclusion

Contributions:

- 1st-order optimization with nonsmooth inner problem
- **Local linear convergence** of the Jacobian
- **Leverage sparsity** to speed up hypergradient computation

Future work:

- Convergence of the bilevel procedure
- Smarter outer solver

References:

- Long version on **arXiv** soon
- Open source package `https://github.com/QB3/sparse-ho`

Bibliographie I

- Agrawal, A. et al. “Differentiable convex optimization layers”. In: NeurIPS. 2019, pp. 9558–9570.
- Akaike, H. “A new look at the statistical model identification”. In: IEEE Trans. Automat. Control AC-19 (1974), pp. 716–723.
- Belloni, A., V. Chernozhukov, and L. Wang. “Square-root Lasso: pivotal recovery of sparse signals via conic programming”. In: Biometrika 98.4 (2011), pp. 791–806.
- Bengio, Y. “Gradient-based optimization of hyperparameters”. In: Neural computation 12.8 (2000), pp. 1889–1900.
- Bergstra, J. and Y. Bengio. “Random search for hyper-parameter optimization”. In: Journal of Machine Learning Research 13.2 (2012).
- Bickel, P. J., Y. Ritov, and A. B. Tsybakov. “Simultaneous analysis of Lasso and Dantzig selector”. In: Ann. Statist. 37.4 (2009), pp. 1705–1732.

Bibliographie II

- Breheny, P. and J. Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection". In: Ann. Appl. Stat. 5.1 (2011), p. 232.
- 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).
- Candès, E. J., M. B. Wakin, and S. P. Boyd. "Enhancing Sparsity by Reweighted l_1 Minimization". In: J. Fourier Anal. Applicat. 14.5-6 (2008), pp. 877–905.
- Diamond, S. and S. Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: Journal of Machine Learning Research 17.83 (2016), pp. 1–5.
- Domke, J. "Generic methods for optimization-based modeling". In: AISTATS. Vol. 22. 2012, pp. 318–326.

Bibliographie III

- Franceschi, L. et al. “Forward and reverse gradient-based hyperparameter optimization”. In: ICML. 2017, pp. 1165–1173.
- Frecon, J., S. Salzo, and M. Pontil. “Bilevel learning of the group lasso structure”. In: NeurIPS. 2018, pp. 8301–8311.
- Larsen, J. et al. “Design and regularization of neural networks: the optimal use of a validation set”. In: Neural Networks for Signal Processing VI. Proceedings of the 1996 IEE 1996.
- Liu, D. C. and J. Nocedal. “On the limited memory BFGS method for large scale optimization”. In: Mathematical programming 45.1-3 (1989), pp. 503–528.
- Nocedal, J. and S. J. Wright. Numerical optimization. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006.

Bibliographie IV

- Obozinski, G., B. Taskar, and M. I. Jordan. "Joint covariate selection and joint subspace selection for multiple classification problems". In: Statistics and Computing 20.2 (2010), pp. 231–252.
- Ochs, P. et al. "Bilevel optimization with nonsmooth lower level problems". In: SSVM. 2015, pp. 654–665.
- 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.
- Pedregosa, F. "Hyperparameter optimization with approximate gradient". In: ICML. Vol. 48. 2016, pp. 737–746.
- Peyré, G. and J. M. Fadili. "Learning analysis sparsity priors". In: Sampta. 2011.
- Schwarz, G. "Estimating the dimension of a model". In: Ann. Statist. 6.2 (1978), pp. 461–464.

Bibliographie V

- Stein, C. M. “Estimation of the mean of a multivariate normal distribution”. In: Ann. Statist. 9.6 (1981), pp. 1135–1151.
- Stone, L. R. A. and J.C. Ramer. “Estimating WAIS IQ from Shipley Scale scores: Another cross-validation”. In: Journal of clinical psychology 21.3 (1965), pp. 297–297.

Proof in the smooth case

$$\arg\min_{\lambda \in \mathbb{R}^r} \left\{ \mathcal{L}(\lambda) := \mathcal{C}\left(\hat{\beta}^{(\lambda)}\right) \right\}$$
$$\text{s.t. } \hat{\beta}^{(\lambda)} \in \arg\min_{\beta \in \mathbb{R}^p} \Phi(\beta, \lambda) \ .$$

then

$$\underbrace{\nabla_\lambda \mathcal{L}(\lambda)}_{\in \mathbb{R}^r} = \underbrace{-\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda)}_{\in \mathbb{R}^{r \times p}} \underbrace{\left(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda)\right)^{-1}}_{\in \mathbb{R}^{p \times p}} \underbrace{\nabla \mathcal{C}(\hat{\beta}^{(\lambda)})}_{\in \mathbb{R}^p} \ . \quad (3)$$

$$\text{Smooth case:} \quad \forall \lambda, \ \nabla_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) = 0 \ , \quad (4)$$

Taking the gradient w.r.t. λ:

$$\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda) + \hat{\mathcal{J}}^\top_{(\lambda)} \nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda) = 0 \ . \quad (5)$$

The Jacobian $\hat{\mathcal{J}}^\top_{(\lambda)}$ is computed by solving the linear system:

$$\hat{\mathcal{J}}^\top_{(\lambda)} = -\underbrace{\nabla^2_{\beta,\lambda} \Phi(\hat{\beta}^{(\lambda)}, \lambda)}_{\in \mathbb{R}^{r \times p}} \underbrace{\left(\nabla^2_\beta \Phi(\hat{\beta}^{(\lambda)}, \lambda)\right)^{-1}}_{\in \mathbb{R}^{p \times p}} \ . \quad (6)$$

Cross Validation fits the framework

CV formulation for the Lasso:

$$\begin{aligned}
&\arg\min_{\lambda \in \mathbb{R}} \mathcal{L}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} \| y^{\mathsf{val}_k} - X^{\mathsf{val}_k} \hat{\beta}^{(\lambda,k)} \|_2^2 \\
&\text{s.t. } \hat{\beta}^{(\lambda,k)} \in \arg\min_{\beta \in \mathbb{R}^p} \tfrac{1}{2n} \| y^{\mathsf{train}_k} - X^{\mathsf{train}_k} \beta \|^2 + \lambda \|\beta\|_1 \ ,
\end{aligned}$$

Consider an extended variable $\beta \in \mathbb{R}^{K \times p}$, to fit cross-validation in a bilevel formulation