Comparing regularisation paths of (conjugate) gradient estimators in ridge regression

Laura Hucker, Markus Reiß, Thomas Stark

StatMathAppli 2025 in Fréjus // September 4, 2025

Supported by
DFG FOR 5381 – Mathematical Statistics in the Information Age

Linear regression setting

Under random design

$$y_i = x_i^\top \beta_0 + \varepsilon_i, \quad i = 1, \ldots, n$$

- $\beta_0 \in \mathbb{R}^p$ unknown true coefficient vector
- i.i.d. observations (x_i, y_i) of $\mathbb{R}^p$-valued feature vectors x_i, $\mathbb{R}$-valued responses y_i
- ε_i error variables, $\mathbb{E}[\varepsilon_i \,|\, x_i] = 0$, $\mathrm{Var}(\varepsilon_i \,|\, x_i) = \sigma^2$, $\sigma > 0$
- $p \gg n$ possible (high-dim. setting)
- **short:** $y = X\beta_0 + \varepsilon$ with $y := (y_1, \ldots, y_n)^\top$, $X := (x_1, \ldots, x_n)^\top$, $\varepsilon := (\varepsilon_1, \ldots, \varepsilon_n)^\top$
- assumptions: $\beta_0 = X^+ X \beta_0$, $\sum_{j=1,\ldots,p:\, s_j = s_i} \langle X^\top y, v_j \rangle^2 > 0$, $i = 1, \ldots, p$

Penalised least squares criterion
= Ridge regression (RR) objective

$$\hat{\beta}_\lambda^{\mathsf{RR}} := \underset{\beta \in \mathbb{R}^p}{\mathsf{argmin}}\, \mathscr{E}_\lambda(\beta) \quad \text{with} \quad \mathscr{E}_\lambda(\beta) := \frac{1}{2n}\|y - X\beta\|^2 + \frac{\lambda}{2}\|\beta\|^2, \quad \lambda \geqslant 0$$

- $\lambda = 0$: $\hat{\beta}_0^{\mathsf{RR}} = X^+ y$ minimum-norm solution of $y = X\beta$, $\hat{\Sigma}_0^{-1} := \hat{\Sigma}^+$ [Ali et al. 2019]
- $\lambda > 0$: $\hat{\beta}_\lambda^{\mathsf{RR}} = \hat{\Sigma}_\lambda^{-1} \cdot n^{-1} X^\top y$, where $\hat{\Sigma}_\lambda := \hat{\Sigma} + \lambda I_p$ with $\hat{\Sigma} := n^{-1} X^\top X$
- for simplicity: assume $\hat{\Sigma}_\lambda$ has p distinct eigenvalues

Penalised least squares criterion

= Ridge regression (RR) objective

$$\mathscr{E}_\lambda(\beta) := \frac{1}{2n}\|y - X\beta\|^2 + \frac{\lambda}{2}\|\beta\|^2$$

$$= \frac{1}{2}\langle \hat{\Sigma}_\lambda \beta, \beta\rangle - \frac{1}{n}\langle X^\top y, \beta\rangle + \frac{1}{2n}\|y\|^2$$

$$= \frac{1}{2}\|\hat{\Sigma}_\lambda^{1/2}\beta - y_\lambda\|^2 + \frac{1}{2n}\|y\|^2 - \frac{1}{2}\|y_\lambda\|^2$$

with

$$y_\lambda := \frac{1}{n}\hat{\Sigma}_\lambda^{-1/2}X^\top y = \hat{\Sigma}_\lambda^{1/2}\beta_\lambda + \varepsilon_\lambda, \quad \beta_\lambda := \hat{\Sigma}_\lambda^{-1}\hat{\Sigma}\beta_0, \quad \varepsilon_\lambda := \frac{1}{n}\hat{\Sigma}_\lambda^{-1/2}X^\top \varepsilon$$

normal equations: $\hat{\Sigma}_\lambda \hat{\beta}_\lambda^{\mathrm{RR}} = \hat{\Sigma}_\lambda^{1/2} y_\lambda$ s.t. $\hat{\beta}_\lambda^{\mathrm{RR}} = \beta_\lambda + \hat{\Sigma}_\lambda^{-1/2}\varepsilon_\lambda$

How to calculate the RR estimator in practice?

API Reference > sklearn.linear_model > Ridge

Ridge

class sklearn.linear_model.**Ridge**(*alpha=1.0*, ***, *fit_intercept=True*, *copy_X=True*, *max_iter=None*, *tol=0.0001*, *solver='auto'*, *positive=False*, *random_state=None*) [source]

Linear least squares with l2 regularization.

Minimizes the objective function:

```
||y - Xw||^2_2 + alpha * ||w||^2_2
```

solver : *{'auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'}, default='auto'*

Solver to use in the computational routines:

- 'auto' chooses the solver automatically based on the type of data.
- 'svd' uses a Singular Value Decomposition of X to compute the Ridge coefficients. It is the most stable solver, in particular more stable for singular matrices than 'cholesky' at the cost of being slower.
- 'cholesky' uses the standard scipy.linalg.solve function to obtain a closed-form solution.
- 'sparse_cg' uses the conjugate gradient solver as found in scipy.sparse.linalg.cg. As an iterative algorithm, this solver is more appropriate than 'cholesky' for large-scale data (possibility to set `tol` and `max_iter`).
- 'lsqr' uses the dedicated regularized least-squares routine scipy.sparse.linalg.lsqr. It is the fastest and uses an iterative procedure.
- 'sag' uses a Stochastic Average Gradient descent, and 'saga' uses its improved, unbiased version named SAGA. Both methods also use an iterative procedure, and are often faster than other solvers when both n_samples and n_features are large. Note that 'sag' and 'saga' fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing.
- 'lbfgs' uses L-BFGS-B algorithm implemented in `scipy.optimize.minimize`. It can be used only when `positive` is True.

How to calculate the RR estimator in practice?

☰ scikit learn

API Reference > sklearn.linear_model > Ridge

Ridge

```
class sklearn.linear_model.Ridge(alpha=1.0, *, fit_intercept=True, copy_X=True, max_iter=None, tol=0.0001, solver='auto', positive=False, random_state=None)
```
[source]

Linear least squares with l2 regularization.

Minimizes the objective function:

```
||y - Xw||^2_2 + alpha * ||w||^2_2
```

solver : *{'auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'}, default='auto'*

Solver to use in the computational routines:

- 'auto' chooses the solver automatically based on the type of data.
- 'svd' uses a Singular Value Decomposition of X to compute the Ridge coefficients. It is the most stable solver, in particular more stable for singular matrices than 'cholesky' at the cost of being slower.
- 'cholesky' uses the standard scipy.linalg.solve function to obtain a closed-form solution.
- 'sparse_cg' uses the conjugate gradient solver as found in scipy.sparse.linalg.cg. As an iterative algorithm, this solver is more appropriate than 'cholesky' for large-scale data (possibility to set `tol` and `max_iter`).
- 'lsqr' uses the dedicated regularized least-squares routine scipy.sparse.linalg.lsqr. It is the fastest and uses an iterative procedure.
- 'sag' uses a Stochastic Average Gradient descent, and 'saga' uses its improved, unbiased version named SAGA. Both methods also use an iterative procedure, and are often faster than other solvers when both n_samples and n_features are large. Note that 'sag' and 'saga' fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing.
- 'lbfgs' uses ... used only w...

By iterative solvers!

Gradient descent/flow (GD/GF)

$\eta > 0$, $k \in \mathbb{N}$, $\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,0} = 0$,

$$
\begin{aligned}
\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k} &:= \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta \nabla \mathscr{E}_\lambda(\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1}) \\
&= \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta(\hat{\Sigma}_\lambda \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \hat{\Sigma}_\lambda^{1/2} y_\lambda)
\end{aligned}
$$

Gradient descent/flow (GD/GF)

$\eta > 0$, $k \in \mathbb{N}$, $\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,0} = 0$,

$$\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k} := \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta \nabla \mathscr{E}_{\lambda}(\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1})$$

$$\left\downarrow {\scriptstyle \eta_k = t/k,\; k \to \infty,\; t > 0} \right. = \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta(\hat{\Sigma}_{\lambda}\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \hat{\Sigma}^{1/2}_{\lambda} y_{\lambda})$$

Gradient descent/flow (GD/GF)

$\eta > 0,\ k \in \mathbb{N},\ \hat\beta^{\mathsf{GD}}_{\lambda,\eta,0} = 0,$

$$\hat\beta^{\mathsf{GD}}_{\lambda,\eta,k} := \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta \nabla \mathscr{E}_\lambda(\hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1})$$

$$= \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta(\hat\Sigma_\lambda \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \hat\Sigma_\lambda^{1/2} y_\lambda)$$

$\eta_k = t/k,$
$k \to \infty,$
$t > 0$

$\hat\beta^{\mathsf{GF}}_{\lambda,t}$ satisfying $\hat\beta^{\mathsf{GF}}_{\lambda,0} = 0,$

$$\frac{d}{dt}\hat\beta^{\mathsf{GF}}_{\lambda,t} = -(\hat\Sigma_\lambda \hat\beta^{\mathsf{GF}}_{\lambda,t} - \hat\Sigma_\lambda^{1/2} y_\lambda),\ t \geqslant 0$$

Gradient descent/flow (GD/GF)

$$\eta > 0,\, k \in \mathbb{N},\, \hat\beta^{\mathsf{GD}}_{\lambda,\eta,0} = 0,$$

$$\hat\beta^{\mathsf{GD}}_{\lambda,\eta,k} := \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta \nabla \mathscr{E}_\lambda(\hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1})$$

$$= \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta(\hat\Sigma_\lambda \hat\beta^{\mathsf{GD}}_{\lambda,\eta,k-1} - \hat\Sigma_\lambda^{1/2} y_\lambda)$$

$\eta_k = t/k,$
$k \to \infty,$
$t > 0$

$$\hat\beta^{\mathsf{GF}}_{\lambda,t} = \hat\Sigma_\lambda^{-1/2}(I_p - R^{\mathsf{GF}}_t(\hat\Sigma_\lambda))y_\lambda,$$

$$R^{\mathsf{GF}}_t(x) = \exp(-tx)$$

Gradient descent/flow (GD/GF)

$$\eta > 0,\ k \in \mathbb{N},\ \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,0} = 0,$$

$$\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k} := \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta \nabla \mathscr{E}_{\lambda}(\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1})$$

$$= \hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \eta(\hat{\Sigma}_{\lambda}\hat{\beta}^{\mathsf{GD}}_{\lambda,\eta,k-1} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})$$

$\eta_k = t/k,$

$k \to \infty,$

$t > 0$

$$\hat{\beta}^{\mathsf{GF}}_{\lambda,t} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_t^{\mathsf{GF}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R_t^{\mathsf{GF}}(x) = \exp(-tx)$$

Conjugate gradients (CG)

Algorithm 1 Penalised CG method

$\hat{\beta}^{\mathrm{CG}}_{\lambda,0} \leftarrow 0,\ q_0 \leftarrow \frac{1}{n}X^{\top}y,\ d_0 \leftarrow q_0,\ e_0 \leftarrow Xd_0,\ k = 1$
while (no division by zero) **do**
    $a_k \leftarrow \|q_{k-1}\|^2/(\frac{1}{n}\|e_{k-1}\|^2 + \lambda\|d_{k-1}\|^2)$
    $\hat{\beta}^{\mathrm{CG}}_{\lambda,k} \leftarrow \hat{\beta}^{\mathrm{CG}}_{\lambda,k-1} + a_k d_{k-1}$
    $q_k \leftarrow q_{k-1} - \frac{a_k}{n}X^{\top}e_{k-1} - \lambda a_k d_{k-1}$
    $b_k \leftarrow \|q_k\|^2/\|q_{k-1}\|^2$
    $d_k \leftarrow q_k + b_k d_{k-1}$
    $e_k \leftarrow Xd_k$
    $k \leftarrow k+1$
end while

Gradient descent/flow (GD/GF)

$$\hat{\beta}^{\mathsf{GF}}_{\lambda,t} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R^{\mathsf{GF}}_t(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R^{\mathsf{GF}}_t(x) = \exp(-tx)$$

Conjugate gradients (CG)

$$\hat{\beta}^{\mathsf{CG}}_{\lambda,k} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R^{\mathsf{CG}}_k(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R^{\mathsf{CG}}_k := \arg\min_{P_k} \|P_k(\hat{\Sigma}_{\lambda})y_{\lambda}\|^2$$

[Phatak and de Hoog 2002]

Gradient descent/flow (GD/GF)

$$\hat{\beta}^{\mathsf{GF}}_{\lambda,t} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R^{\mathsf{GF}}_t(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R^{\mathsf{GF}}_t(x) = \exp(-tx)$$

Conjugate gradients (CG)

$$\hat{\beta}^{\mathsf{CG}}_{\lambda,k} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R^{\mathsf{CG}}_k(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R^{\mathsf{CG}}_k := \arg\min_{P_k} \|P_k(\hat{\Sigma}_{\lambda})y_{\lambda}\|^2$$

[Phatak and de Hoog 2002]

$$\Rightarrow \quad \hat{\beta}^{\mathsf{CG}}_{\lambda,p} = \hat{\beta}^{\mathsf{RR}}_{\lambda}$$

Gradient descent/flow (GD/GF)

$$\hat{\beta}^{\mathsf{GF}}_{\lambda,t} = \hat{\Sigma}_\lambda^{-1/2}(I_p - R_t^{\mathsf{GF}}(\hat{\Sigma}_\lambda))y_\lambda,$$

$$R_t^{\mathsf{GF}}(x) = \exp(-tx)$$

Conjugate gradients (CG)

continuous iteration path:

$$t = k + \alpha,\, k \in [\![0,p-1]\!],\, \alpha \in (0,1],$$

$$\hat{\beta}^{\mathsf{CG}}_{\lambda,t} := \hat{\Sigma}_\lambda^{-1/2}(I_p - R_t^{\mathsf{CG}}(\hat{\Sigma}_\lambda))y_\lambda,$$

$$R_t^{\mathsf{CG}} := (1-\alpha)R_k^{\mathsf{CG}} + \alpha R_{k+1}^{\mathsf{CG}}$$

[Hucker and Reiß 2025]

Error analysis

For in-sample prediction errors

- pure prediction error: $n^{-1}\|X(\hat{\beta}-\beta_0)\|^2 = \|\hat{\Sigma}^{1/2}(\hat{\beta}-\beta_0)\|^2$
- population risk:
  $\beta \mapsto \mathbb{E}[\mathscr{E}_\lambda(\beta)\,|\,X] = \frac{1}{2}\|\hat{\Sigma}_\lambda^{1/2}(\beta-\beta_\lambda)\|^2 + \frac{1}{2}\sigma^2 + \frac{1}{2}\|\hat{\Sigma}^{1/2}\beta_0\|^2 - \frac{1}{2}\|\hat{\Sigma}_\lambda^{1/2}\beta_\lambda\|^2$
- excess penalised prediction error:
  $\|\hat{\Sigma}_\lambda^{1/2}(\hat{\beta}-\beta_\lambda)\|^2 = n^{-1}\|X(\hat{\beta}-\beta_\lambda)\|^2 + \lambda\|\hat{\beta}-\beta_\lambda\|^2 =: \ell^{\text{in}}_{\lambda,\beta_\lambda}(\hat{\beta})$
- regularised in-sample prediction loss:
  $\ell^{\text{in}}_{\lambda,\gamma}(\hat{\beta}) := \|\hat{\Sigma}_\lambda^{1/2}(\hat{\beta}-\gamma)\|^2 = n^{-1}\|X(\hat{\beta}-\gamma)\|^2 + \lambda\|\hat{\beta}-\gamma\|^2$
- prediction risk: $\mathscr{R}^{\text{in}}_{\lambda,\gamma}(\hat{\beta}) := \mathbb{E}[\ell^{\text{in}}_{\lambda,\gamma}(\hat{\beta})\,|\,X]$

Standard prediction error decomposition
(Useful for linear regularisation methods)

Consider for any residual filter function $R : [0,\infty) \to \mathbb{R}$ the estimator $\hat{\beta} = \hat{\Sigma}_\lambda^{-1/2}(I_p - R(\hat{\Sigma}_\lambda))y_\lambda$.

Then the penalised prediction loss satisfies $\ell^{\text{in}}_{\lambda,\gamma}(\hat{\beta}) = A_{\lambda,\gamma}(R) + S_\lambda(R) - 2C_{\lambda,\gamma}(R)$

with approximation, stochastic and cross term errors

$$
\begin{aligned}
A_{\lambda,\gamma}(R) &= \|\hat{\Sigma}_\lambda^{1/2}(R(\hat{\Sigma}_\lambda)\beta_\lambda + (\gamma - \beta_\lambda))\|^2, \\
S_\lambda(R) &= \|(I_p - R(\hat{\Sigma}_\lambda))\varepsilon_\lambda\|^2, \\
C_{\lambda,\gamma}(R) &= \langle \hat{\Sigma}_\lambda^{1/2}(R(\hat{\Sigma}_\lambda)\beta_\lambda + (\gamma - \beta_\lambda)), (I_p - R(\hat{\Sigma}_\lambda))\varepsilon_\lambda \rangle \,.
\end{aligned}
$$

If R is deterministic, then $\mathscr{R}^{\text{in}}_{\lambda,\gamma}(\hat{\beta}) = A_{\lambda,\gamma}(R) + \frac{\sigma^2}{n}\text{trace}((I_p - R(\hat{\Sigma}_\lambda))^2\hat{\Sigma}_\lambda^{-1}\hat{\Sigma})$.

Prediction error decomposition for CG

[Hucker and Reiß 2025, Proposition 5.2]

- denote by $x_{1,t}$ the smallest zero on $[0,\infty)$ of R_t^{CG}, $t \in (0,p]$
- for $x \geqslant 0$, $R_{t,<}^{\mathrm{CG}}(x) := R_t^{\mathrm{CG}}(x)\mathbf{1}(x < x_{1,t})$, $R_{t,>}^{\mathrm{CG}}(x) := R_t^{\mathrm{CG}}(x)\mathbf{1}(x > x_{1,t})$

At $t \in [0,p]$, the penalised CG estimator satisfies for $\gamma = \beta_\lambda$

$$\ell^{\mathrm{in}}_{\lambda,\beta_\lambda}(\hat\beta^{\mathrm{CG}}_{\lambda,t}) = A^{\mathrm{CG}}_{\lambda,\beta_\lambda,t} + S^{\mathrm{CG}}_{\lambda,t} - 2C^{\mathrm{CG}}_{\lambda,\beta_\lambda,t} \leqslant 2A^{\mathrm{CG}}_{\lambda,\beta_\lambda,t} + 2S^{\mathrm{CG}}_{\lambda,t}$$

with approximation, stochastic and cross term errors given by

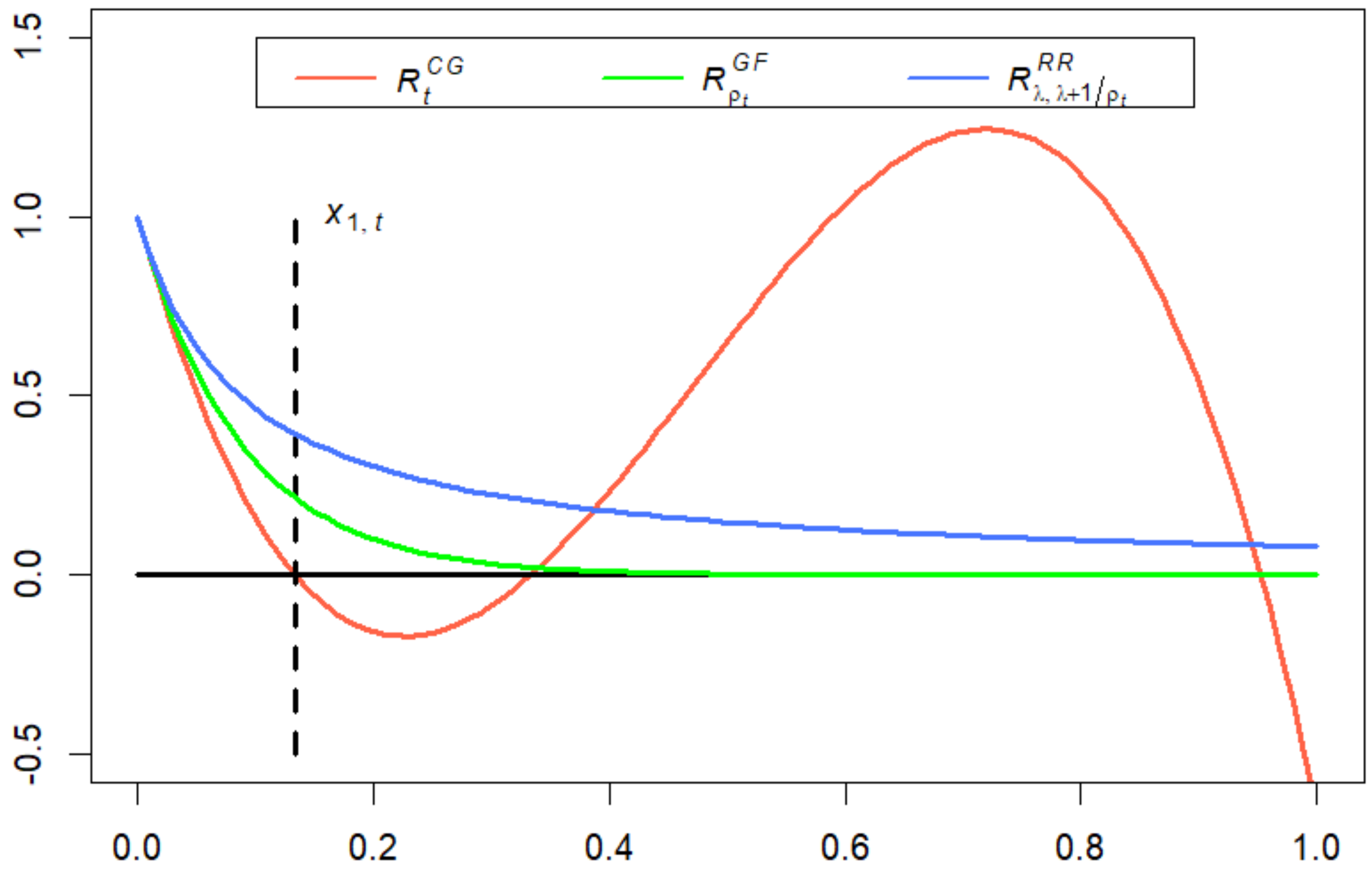
$$A^{\mathrm{CG}}_{\lambda,\beta_\lambda,t} := \|\hat\Sigma_\lambda^{1/2} R^{\mathrm{CG}}_{t,<}(\hat\Sigma_\lambda)^{1/2}\beta_\lambda\|^2 + \|R^{\mathrm{CG}}_t(\hat\Sigma_\lambda)y_\lambda\|^2 - \|R^{\mathrm{CG}}_{t,<}(\hat\Sigma_\lambda)^{1/2}y_\lambda\|^2 \leqslant \|\hat\Sigma_\lambda^{1/2} R^{\mathrm{CG}}_{t,<}(\hat\Sigma_\lambda)^{1/2}\beta_\lambda\|^2,$$

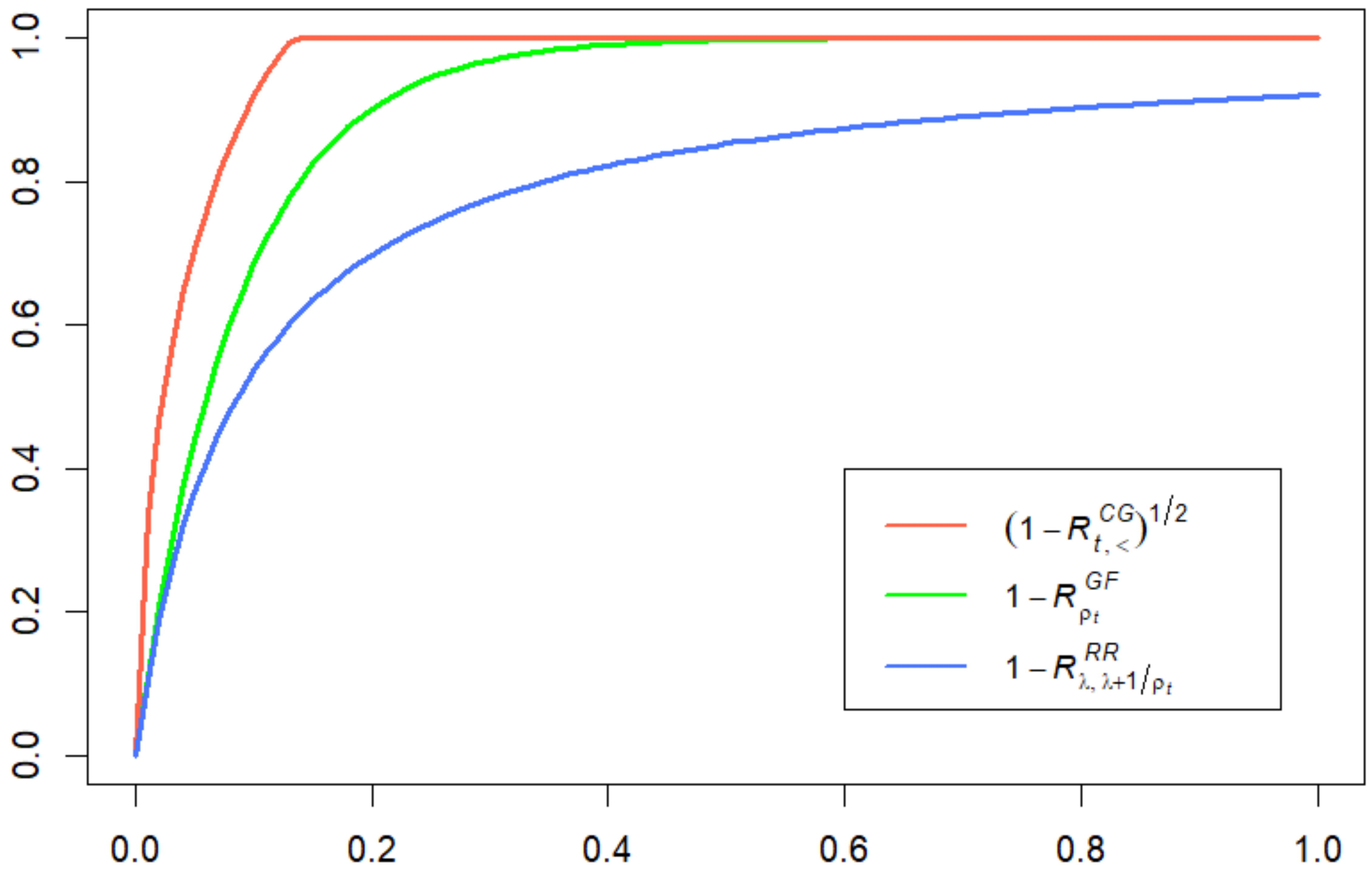
$$S^{\mathrm{CG}}_{\lambda,t} := \|(I_p - R^{\mathrm{CG}}_{t,<}(\hat\Sigma_\lambda))^{1/2}\varepsilon_\lambda\|^2, \quad C^{\mathrm{CG}}_{\lambda,\beta_\lambda,t} := \langle R^{\mathrm{CG}}_{t,>}(\hat\Sigma_\lambda)y_\lambda, \varepsilon_\lambda\rangle\,.$$

$$\rho_t := |(R_t^{\mathrm{CG}})'(0)|,$$

$$(1-\rho_t x)_+ \leqslant R_t^{\mathrm{CG}}(x) \leqslant \exp(-\rho_t)x, \quad x \in [0, x_{1,t}]$$

[Hucker and Reiß 2025, Lemma 4.7]





$$\rho_t := |(R_t^{\mathsf{CG}})'(0)|,$$

$$(1-\rho_t x)_+ \leqslant R_t^{\mathsf{CG}}(x) \leqslant \exp(-\rho_t)x, \quad x \in [0, x_{1,t}]$$

[Hucker and Reiß 2025, Lemma 4.7]

Let $t \in [0,p]$. Assume that $\gamma \in \mathbb{R}^p$ satisfies the condition (CROSS)

$$\langle \hat{\Sigma}_\lambda \exp(-\rho_t \hat{\Sigma}_\lambda/2)\beta_\lambda, \gamma - \beta_\lambda \rangle \geqslant 0.$$

Then the regularised in-sample prediction loss for the CG estimator is bounded by

$$\ell^{\mathsf{in}}_{\lambda,\gamma}(\hat{\beta}^{\mathsf{CG}}_{\lambda,t}) \leqslant 4\overline{A}^{\mathsf{CG}}_{\lambda,\gamma,t} + 2\overline{S}^{\mathsf{CG}}_{\lambda,t}$$

with γ-dependent approximation and stochastic error bounds

$$\overline{A}^{\mathsf{CG}}_{\lambda,\gamma,t} := \|\hat{\Sigma}_\lambda^{1/2}(\beta_\lambda - \exp(-\rho_t\hat{\Sigma}_\lambda/2)\beta_\lambda - \gamma)\|^2, \quad \overline{S}^{\mathsf{CG}}_{\lambda,t} := \|(\rho_t\hat{\Sigma}_\lambda \wedge 1)^{1/2}\varepsilon_\lambda\|^2.$$

Main result

Compare CG and GF errors up to a random time shift

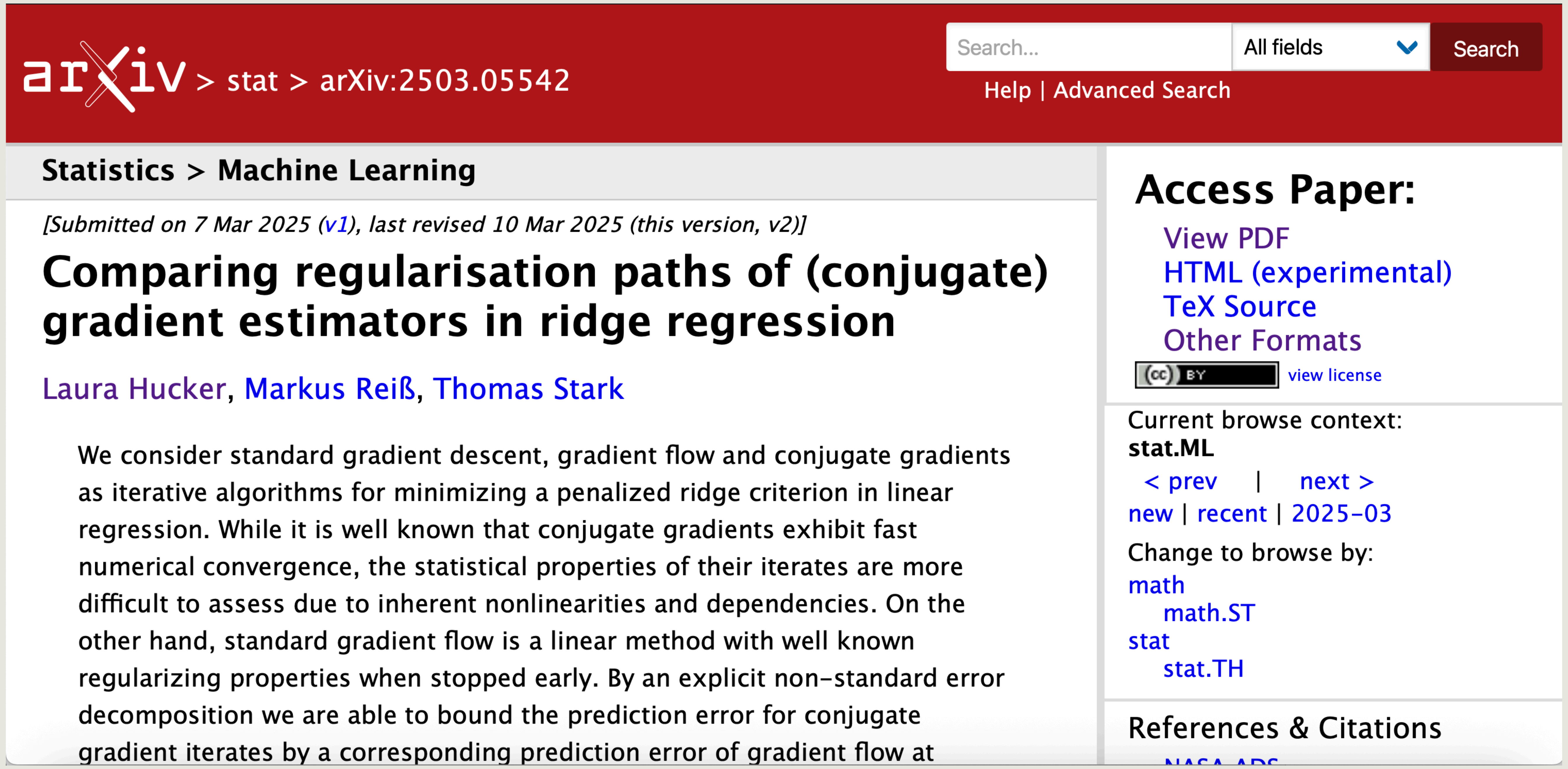
- inverting $t \mapsto \rho_t/2$, introduce the random times
  $\tau_t :=$ inf$\{\tilde{t} \in [0,p] \,|\, |(R^{\mathrm{CG}}_{\tilde{t}})'(0)| \geqslant 2t\} \wedge p, \quad t \geqslant 0$
- under (CROSS), $\mathscr{R}^{\mathrm{in}}_{\lambda,\gamma}(\hat{\beta}^{\mathrm{CG}}_{\lambda,\tau_t}) \leqslant 4A_{\lambda,\gamma}(R^{\mathrm{GF}}_t) + \frac{2\sigma^2}{n}\mathrm{trace}((2t \wedge \hat{\Sigma}_\lambda^{-1})\hat{\Sigma})$
- remains to compare stochastic errors

Let $\mathscr{L}_\lambda(t) := \mathrm{trace}(\exp(-t\hat{\Sigma}_\lambda))$ for $t \geqslant 0$. If $\hat{\Sigma}_\lambda > 0$, then for $t_0 > 0$

$$C_{0,\lambda} := C_{0,\lambda}(\mathscr{L}_\lambda(t_0)) \leqslant \frac{2\,|\,\mathscr{L}_\lambda'(t_0)\,|}{t_0 \mathscr{L}_\lambda''(t_0)} \in (0,\infty)$$

holds, and under (CROSS) for γ we have

$$\forall t \geqslant t_0 : \quad \mathscr{R}^{\mathrm{in}}_{\lambda,\gamma}(\hat{\beta}^{\mathrm{CG}}_{\lambda,\tau_t}) \leqslant \frac{2(1 + C_{0,\lambda})}{1 - \exp(-1/2)} \mathscr{R}^{\mathrm{in}}_{\lambda,\gamma}(\hat{\beta}^{\mathrm{GF}}_{\lambda,t}) \,.$$



arXiv > stat > arXiv:2503.05542

Search... All fields Search

Help | Advanced Search

Statistics > Machine Learning

[Submitted on 7 Mar 2025 (v1), last revised 10 Mar 2025 (this version, v2)]

Comparing regularisation paths of (conjugate) gradient estimators in ridge regression

Laura Hucker, Markus Reiß, Thomas Stark

We consider standard gradient descent, gradient flow and conjugate gradients as iterative algorithms for minimizing a penalized ridge criterion in linear regression. While it is well known that conjugate gradients exhibit fast numerical convergence, the statistical properties of their iterates are more difficult to assess due to inherent nonlinearities and dependencies. On the other hand, standard gradient flow is a linear method with well known regularizing properties when stopped early. By an explicit non-standard error decomposition we are able to bound the prediction error for conjugate gradient iterates by a corresponding prediction error of gradient flow at

Access Paper:

View PDF

HTML (experimental)

TeX Source

Other Formats

(cc) BY view license

Current browse context:

stat.ML

< prev | next >

new | recent | 2025-03

Change to browse by:

math

math.ST

stat

stat.TH

References & Citations

L. Hucker, M. Reiß, T. Stark (2025+). Comparing regularisation paths of (conjugate) gradient estimators in ridge regression. arXiv:2503.05542 [stat.ML]

L. Hucker, M. Reiß (2025). Early stopping for conjugate gradients in statistical inverse problems. *Numer. Math. (to appear)*. arXiv:2406.15001 [math.ST]

L. Hucker, M. Wahl (2023). A note on the prediction error of principal component regression in high dimensions. *Theory Probab. Math. Stat.* 109, 37–53. doi:10.1090/tpms/1196

Thanks a lot for your attention!

Some further references

A. Ali, J.Z. Kolter, R.J. Tibshirani (2019). A continuous-time view of early stopping for least squares regression. In: K. Chaudhuri, M. Sugiyama (eds.) Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics. *Proceedings of Machine Learning Research,* vol. 89, pp. 1370–1378

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12(85), 2825–2830

A. Phatak, F. de Hoog (2002). Exploiting the connection between PLS, Lanczos methods and conjugate gradients: alternative proofs of some properties of PLS. *J. Chemom.* 16(7), 361–367. doi:10.1002/cem.728