Computational algorithm for Lasso

can use a very generic coordinate descent algorithm (not gradient descent)

motivation of the algorithm:
consider the objective function and the corresponding Karush-Kuhn-Tucker (KKT) conditions by taking the sub-differential:

$$
\begin{aligned}
& \frac{\partial}{\partial j}(\|Y - X\beta\|_2^2/n + \lambda\|\beta\|_1) \\
= \quad & G_j(\beta) + \lambda e_j, \\
& G(\beta) = -2X^T(Y - X\beta)/n, \\
& e_j = \text{sign}(\beta_j) \text{ if } \beta_j \neq 0, \quad e_j \in [-1, 1] \text{ if } \beta_j = 0
\end{aligned}
$$

from convex optimization:
solution is characterized by

$$0 \in \text{ sub-differential at}(\hat{\beta})$$

this implies the KKT-conditions (Lemma 2.1, Bühlmann and van de Geer (2011):

$$G_j(\hat{\beta}) = -\text{sign}(\hat{\beta}_j)\lambda \text{ if } \hat{\beta}_j \neq 0,$$
$$|G_j(\hat{\beta})| \leq \lambda \text{ if } \hat{\beta}_j = 0.$$

an interesting characterization of the Lasso solution!

in abbreviated form:

1: Let $\beta^{[0]} \in \mathbb{R}^p$ be an initial parameter vector. For $m = 1, 2, \ldots$

2: **repeat**

3: Proceed componentwise $j = 1, 2, \ldots, p, 1, 2, \ldots p, 1, 2, \ldots$ update:

$$\text{if } |G_j(\underbrace{\beta_{-j}^{[m-1]}}_{\text{prev. parameter with } j\text{th comp}=0})| \le \lambda : \text{ set } \beta_j^{[m]} = 0,$$

otherwise: $\beta_j^{[m]}$ is the minimizer of the objective function with respect to the jth component but keeping all others fixed

4: **until** numerical convergence

1: Let $\beta^{[0]} \in \mathbb{R}^p$ be an initial parameter vector. Set $m = 0$.

2: **repeat**

3: Increase m by one: $m \leftarrow m + 1$.
Denote by $\mathcal{S}^{[m]}$ the index cycling through the coordinates $\{1, \ldots, p\}$:
$\mathcal{S}^{[m]} = \mathcal{S}^{[m-1]} + 1 \mod p$. Abbreviate by $j = \mathcal{S}^{[m]}$ the value of $\mathcal{S}^{[m]}$.

4: if $|G_j(\beta^{[m-1]}_{-j})| \leq \lambda:$ set $\beta^{[m]}_j = 0$,
otherwise: $\beta^{[m]}_j = \text{argmin}_{\beta_j} Q_\lambda(\beta^{[m-1]}_{+j})$,
where $\beta^{[m-1]}_{-j}$ is the parameter vector where the jth component is set to zero and $\beta^{[m-1]}_{+j}$ is the parameter vector which equals $\beta^{[m-1]}$ except for the jth component where it is equal to β_j (i.e. the argument we minimize over).

5: **until** numerical convergence

for the squared error loss: the update in Step 4 is explicit (a soft-thresholding operation)

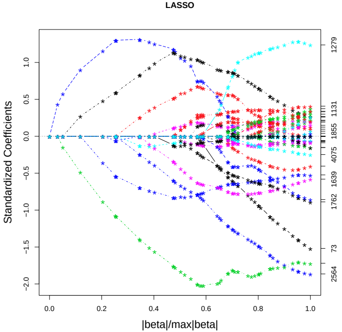
active set strategy can speed up the algorithm for sparse cases: mainly work on the non-zero coordinates and up-date all coordinates e.g. every 20th times

`R`-package `glmnet`

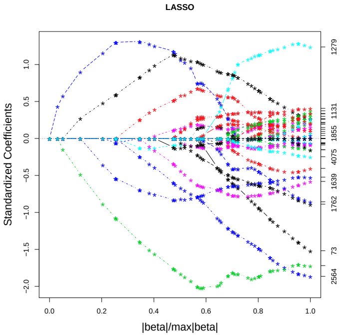
The Lasso regularization path

compute $\hat{\beta}(\lambda)$ over "all" λ

- just a grid of λ-values and interpolate linearly (the true solution path over all λ is piecewise linear)
- for $\lambda_{\max} = \max_j |(2X^T Y/n)_j|$: $\hat{\beta}(\lambda_{\max}) = 0$
  (because of KKT conditions!)



plot against $\|\hat{\beta}(\lambda)\|_1 / \max_\lambda \|\hat{\beta}(\lambda)\|_1$ (λ small is to the right)



regularization path: in general, “not monotone in the non-zeros”
it can happen in general that e.g.

$$\hat{\beta}_j(\lambda) \neq 0,\ \hat{\beta}_j(\lambda') = 0 \text{ for } \lambda' < \lambda$$

Generalized linear models (GLMs)

univariate response Y, covariate $X \in \mathcal{X} \subseteq \mathbb{R}^p$

GLM: $Y_1, \ldots, Y_n$ independent

$$g(\mathbb{E}[Y_i|X_i = x]) = \underbrace{\mu + \sum_{j=1}^{p} \beta_j x^{(j)}}_{=f(x)=f_{\mu,\beta}(x)}$$

$g(\cdot)$ real-valued, known link function
μ an intercept term: the intercept is important: we cannot simply center the response and ignore an intercept...

Lasso: defined as ℓ_1-norm penalized negative log-likelihood (where μ is not penalized)

software: `glmnet` in `R`

Example: logistic (penalized) regression
$Y \in \{0,1\}$
$\pi(x) = \mathbb{E}[Y|X=x] = \mathbb{P}[Y=1|X=x]$
logistic link function: $g(\pi) = \log(\pi/(1-\pi))$ $(\pi \in (0,1))$

denote by $\pi_i = \mathbb{P}[Y_1 = 1|X_i]$
$\log(\pi_i/(1-\pi_i)) = \mu + X_i^T\beta,\ \pi_i = \frac{\exp(\mu+X_i^T\beta)}{1+\exp(\mu+X_i^T\beta)}$

log-likelihood

$$\begin{aligned} & \sum_{i=1}^{n} \log(\pi_i^{Y_i}(1-\pi_i)^{1-Y_i}) = \sum_{i=1}^{n}(Y_i \log(\pi_i) + (1-Y_i)\log(1-\pi_i)) \\ = & \sum_{i=1}^{n}(Y_i \underbrace{\log(\pi_i/(1-\pi_i))}_{\mu+X_i^T\beta} + \underbrace{\log(1-\pi_i)}_{\log(1+\exp(\mu+X_i^T\beta))}) \end{aligned}$$

negative log-likelihood

$$-\ell(\mu, \beta) = \sum_{i=1}^{n} (-Y_i(\mu + X_i^T \beta) + \log(1 + \exp(\mu + X_i^T \beta)))$$

which is a convex function in μ, β

Lasso for linear logistic regression:

$$\hat{\mu}, \hat{\beta} = \operatorname{argmin}_{\mu,\beta}(-\ell(\mu, \beta) + \lambda \|\beta\|_1)$$

μ is not penalized

note: often used nowadays for classification with deep neural networks

$$\log(\pi_i/(1-\pi_i)) = \mu + \underbrace{X^T\beta^{(1)}}_{\text{NN with linear connection}} + \underbrace{W_\theta(X)^T}_{\text{features from last NN layer}} \beta^{(2)}$$

estimator:

$$\hat{\mu}, \hat{\beta}^{(1)}, \hat{\beta}^{(2)}, \hat{\theta} = \text{argmin} - \ell\left(\mu, \beta^{(1)}, \beta^{(2)}, \theta\right) + \lambda(\|\beta^{(1)}\|_1 + \|\beta^{(2)}\|_1)$$

this is now a highly non-convex function in θ...!

if somebody gives you the feature mapping $w_\theta(\cdot)$ (e.g. trained on large image database), then one can use logistic Lasso

IV. Group Lasso (... continued after material from visualizer)

Parameterization of model matrix

4 levels, $p = 2$ variables

main effects only

```
> xx1
[1] 0 1 2 3 3 2 1 0
Levels: 0 1 2 3
> xx2
[1] 3 3 2 2 1 1 0 0
Levels: 0 1 2 3

> model.matrix(~xx1+xx2,
contrasts=list(xx1="contr.sum",xx2="contr.sum"))
  (Intercept) xx11 xx12 xx13 xx21 xx22 xx23
1           1    1    0    0   -1   -1   -1
2           1    0    1    0   -1   -1   -1
3           1    0    0    1    0    0    1
4           1   -1   -1   -1    0    0    1
5           1   -1   -1   -1    0    1    0
6           1    0    0    1    0    1    0
7           1    0    1    0    1    0    0
8           1    1    0    0    1    0    0
attr(,"assign")
[1] 0 1 1 1 2 2 2
attr(,"contrasts")
attr(,"contrasts")$xx1
[1] "contr.sum"

attr(,"contrasts")$xx2
[1] "contr.sum"
```

with interaction terms

```
> model.matrix(~xx1*xx2,
contrasts=list(xx1="contr.sum",xx2="contr.sum"))
  (Intercept) xx11 xx12 xx13 xx21 xx22 xx23 xx11:xx21 xx12:xx21 xx13:xx21
1           1    1    0    0   -1   -1   -1        -1         0         0
2           1    0    1    0   -1   -1   -1         0        -1         0
3           1    0    0    1    0    0    1         0         0         0
4           1   -1   -1   -1    0    0    1         0         0         0
5           1   -1   -1   -1    0    1    0         0         0         0
6           1    0    0    1    0    1    0         0         0         0
7           1    0    1    0    1    0    0         0         1         0
8           1    1    0    0    1    0    0         1         0         0
  xx11:xx22 xx12:xx22 xx13:xx22 xx11:xx23 xx12:xx23 xx13:xx23
1        -1         0         0        -1         0         0
2         0        -1         0         0        -1         0
3         0         0         0         0         0         1
4         0         0         0        -1        -1        -1
5        -1        -1        -1         0         0         0
6         0         0         1         0         0         0
7         0         0         0         0         0         0
8         0         0         0         0         0         0
attr(,"assign")
 [1] 0 1 1 1 2 2 2 3 3 3 3 3 3 3 3 3
attr(,"contrasts")
attr(,"contrasts")$xx1
[1] "contr.sum"

attr(,"contrasts")$xx2
[1] "contr.sum"
```

Prediction of DNA splice sites (Ch. 4.3.1 in Bühlmann and van de Geer (2011))

want to predict donor splice site where coding and non-coding regions in DNA start/end

$$\underbrace{\cdot\ \cdot\ \cdot\ \cdot}_{\text{exon: coding}} \qquad GT \qquad \underbrace{\cdot\ \cdot\ \cdot}_{\text{intron: non-coding}}$$

seven positions around “GT”

training data:

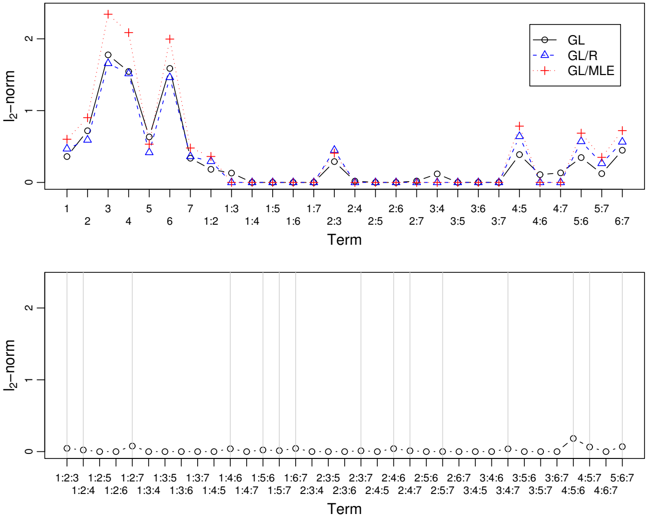
$$\begin{aligned} &Y_i \in \{0,1\} \text{ true donor site or not} \\ &X_i \in \{A, C, G, T\}^7 \text{ positions} \\ &i = 1, \ldots, n \approx 188'000 \end{aligned}$$

unbalanced: $Y_i = 1$: 8415; $Y_i = 0$: 179’438

model: logistic linear regression model with intercept, main effects and interactions up to order 2 (3 variables interact)
$\leadsto$ dimension = 1155

methods:

- Group Lasso
- MLE on $\hat{S} = \{j;\ \hat{\beta}_{\mathcal{G}_j} \neq 0\}$
- as above but with Ridge regularized MLE on $\hat{S}$



mainly main effects (quite debated in computational biology...)

Theoretical guarantees for Group Lasso

follows “similarly” but with more complicated arguments as for the Lasso

Algorithm for Group Lasso

consider the KKT conditions for the objective function

$$Q_\lambda(\beta) = \underbrace{n^{-1}\sum_{i=1}^{n}\rho_\beta(X_i, Y_i)}_{\text{e.g. } \|Y - X\beta\|_2^2/n} + \lambda\sum_{j=1}^{q} m_j\|\beta_{\mathcal{G}_j}\|_2$$

Lemma (Lemma 4.3 in Bühlmann and van de Geer (2011))
Assume $\rho_\beta = n^{-1}\sum_{i=1}^{n}\rho_\beta(X_i, Y_i)$ is differentiable and convex (in β). Then, a necessary and sufficient condition for $\hat{\beta}$ to be a solution is

$$\nabla\rho(\hat{\beta})_{\mathcal{G}_j} = -\lambda m_j \frac{\hat{\beta}_{\mathcal{G}_j}}{\|\hat{\beta}_{\mathcal{G}_j}\|_2} \qquad \text{if } \hat{\beta}_{\mathcal{G}_j} \not\equiv 0,$$

$$\|\nabla\rho(\hat{\beta})_{\mathcal{G}_j}\|_2 \le \lambda m_j \qquad \text{if } \hat{\beta}_{\mathcal{G}_j} \equiv 0$$

block coordinate descent

Algorithm 1 Block Coordinate Descent Algorithm

1: Let $\beta^{[0]} \in \mathbb{R}^p$ be an initial parameter vector. Set $m = 0$.

2: **repeat**

3: Increase m by one: $m \leftarrow m + 1$.
Denote by $\mathscr{S}^{[m]}$ the index cycling through the block coordinates $\{1, \ldots, q\}$:
$\mathscr{S}^{[m]} = \mathscr{S}^{[m-1]} + 1 \bmod q$. Abbreviate by $j = \mathscr{S}^{[m]}$ the value of $\mathscr{S}^{[m]}$.

4: if $\|(-\nabla\rho(\beta^{[m-1]}_{-\mathscr{G}_j})_{\mathscr{G}_j}\|_2 \leq \lambda m_j$: set $\beta^{[m]}_{\mathscr{G}_j} = 0$,
otherwise: $\beta^{[m]}_{\mathscr{G}_j} = \arg\min_{\beta_{\mathscr{G}_j}} Q_\lambda(\beta^{[m-1]}_{+\mathscr{G}_j})$,
where $\beta^{[m-1]}_{-\mathscr{G}_j}$ is defined in (4.14) and $\beta^{[m-1]}_{+\mathscr{G}_j}$ is the parameter vector which equals $\beta^{[m-1]}$ except for the components corresponding to group $\mathscr{G}_j$ whose entries are equal to $\beta_{\mathscr{G}_j}$ (i.e. the argument we minimize over).

5: **until** numerical convergence

block-updates where the blocks correspond to the groups

The generalized Group Lasso penalty

Chapter 4.5 in Bühlmann and van de Geer (2011)

$$\mathrm{pen}(\beta) = \lambda \sum_{j=1}^{q} m_j \sqrt{\beta_{\mathcal{G}_j}^T A_j \beta_{\mathcal{G}_j}},$$

A_j positive definite

can do the computation with standard group Lasso by transformation:

$$\tilde{\beta}_{\mathcal{G}_j} = A_j^{1/2} \beta_{\mathcal{G}_j} \leadsto \mathrm{pen}(\tilde{\beta}) = \lambda \sum_{j=1}^{q} m_j \|\tilde{\beta}_{\mathcal{G}_j}\|_2$$

$$X\beta = \sum_{j=1}^{q} \tilde{X}_{\mathcal{G}_j} \tilde{\beta}_{\mathcal{G}_j} =: \tilde{X}\tilde{\beta},\ \tilde{X}_{\mathcal{G}_j} = X_{\mathcal{G}_j} A_j^{-1/2}$$

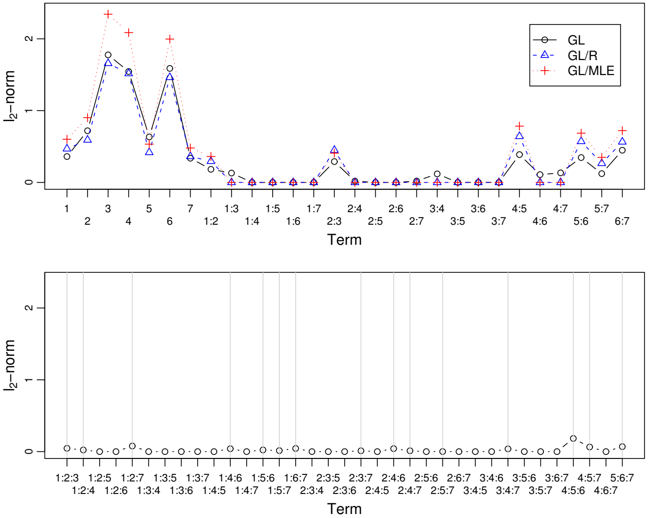
can simply solve the “tilde” problem: $\leadsto \hat{\tilde{\beta}} \leadsto \hat{\beta}_{\mathcal{G}_j} = A_j^{-1/2} \hat{\tilde{\beta}}_{\mathcal{G}_j}$

special but important case: groupwise prediction penalty

$$\mathrm{pen}(\beta) = \sum_{j=1}^{q} m_j \|X_{\mathcal{G}_j}\beta_{\mathcal{G}_j}\|_2 = \lambda \sum_{j=1}^{q} m_j \sqrt{\beta_{\mathcal{G}_j}^T X_{\mathcal{G}_j}^T X_{\mathcal{G}_j}\beta_{\mathcal{G}_j}}$$

$X_{\mathcal{G}_j}^T X_{\mathcal{G}_j}$ typically positive definite for $|\mathcal{G}_j| < n$

- penalty is invariant under arbitrary reparameterizations within every group $\mathcal{G}_j$: important!
- when using an orthogonal parameterization such that $X_{\mathcal{G}_j}^T X_{\mathcal{G}_j} = I$: it is the standard Group Lasso
  with categorical variables: this is in fact what one has in mind (can use groupwise orthogonalized design) or one should use the groupwise prediction penalty



is with groupwise orthogonalized design matrices

High-dimensional additive models

the special case with natural cubic splines

(Ch. 5.3.2 in Bühlmann and van de Geer (2011))

consider the estimation problem wit the SSP penalty:

$$\hat{f}_1, \ldots, \hat{f}_p = \mathrm{argmin}_{f_1,\ldots,f_p \in \mathcal{F}} \Big(\| Y - \sum_{j=1}^{p} f_j \|_n^2 + \lambda_1 \| f_j \|_n + \lambda_2 I(f_j) \Big)$$

where $\mathcal{F}$ = Sobolev space of functions on $[a, b]$ that are continuously differentiable with square integrable second derivatives

Proposition 5.1 in Bühlmann and van de Geer (2011)
Let $a, b \in \mathbb{R}$ such that $a < \min_{i,j}(X_i^{(j)})$ and $b > \max_{i,j}(X_i^{(j)})$. Let $\mathcal{F}$ be as above. Then, the $\hat{f}_j$'s are natural cubic splines with knots at $X_i^{(j)},\ i = 1, \ldots, n$.

implication: the optimization over functions is exactly representable as a parametric problem with dim $\approx 3np$

the optimization over functions is exactly representable as a parametric problem with

therefore:

$$f_j = H_j\beta_j,\ H_j \text{ from natural cubic spline basis}$$

$$\|f_j\|_n = \|H_j\beta_j\|_2/\sqrt{n} = \sqrt{\beta_j^T H_j^T H_j\beta_j}/\sqrt{n}$$

$$I(f_j) = \sqrt{\int ((H_j\beta_j)'')^2} = \sqrt{\beta_j^T \underbrace{(H_j'')^T H_j''}_{=:W_j} \beta} = \sqrt{\beta_j^T W_j\beta_j}$$

$\rightsquigarrow$ convex problem

$$\hat{\beta} = \operatorname{argmin}_\beta \left(\|Y - H\beta\|_2^2/n + \lambda_1 \sum_{j=1}^p \sqrt{\beta_j^T H_j^T H_j\beta_j/n} + \lambda_2 \sum_{j=1}^p \sqrt{\beta_j^T W_j\beta_j} \right)$$

SSS penalty of group Lasso type

for easier computation: instead of

$$\text{SSP penalty} = \lambda_1 \sum_j \|f_j\|_n + \lambda_2 \sum_j I(fj)$$

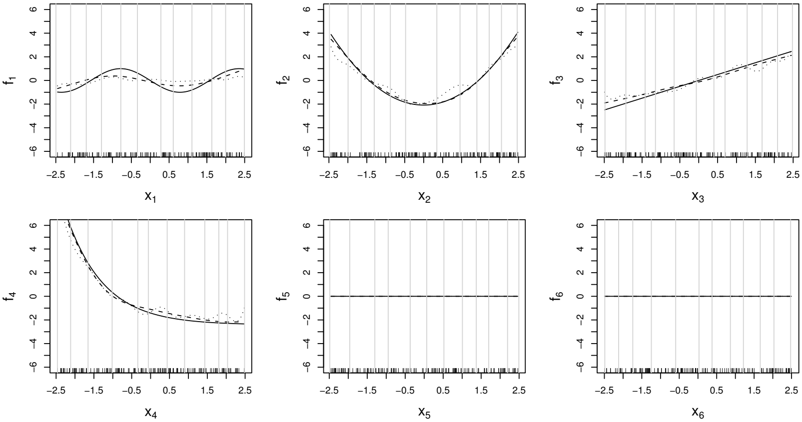
one can also use as an alternative:

$$\text{SSP Group Lasso penalty} = \lambda_1 \sum_j \sqrt{\|f_j\|_n^2 + \lambda_2 I^2(f_j)}$$

in parameterized form, the latter becomes:

$$\lambda_1 \sum_{j=1}^{p} \sqrt{\|H_j\beta_j\|_2^2/n + \lambda_2^2 \beta_j^T W_j \beta_j} = \lambda_1 \sum_{j=1}^{p} \sqrt{\beta_j^T (H_j^T H_j/n + \lambda_2^2 W_j)\beta_j}$$

$\leadsto$ for every λ_2: a generalized Group Lasso penalty simulated example: $n = 150, p = 200$ and 4 active variables

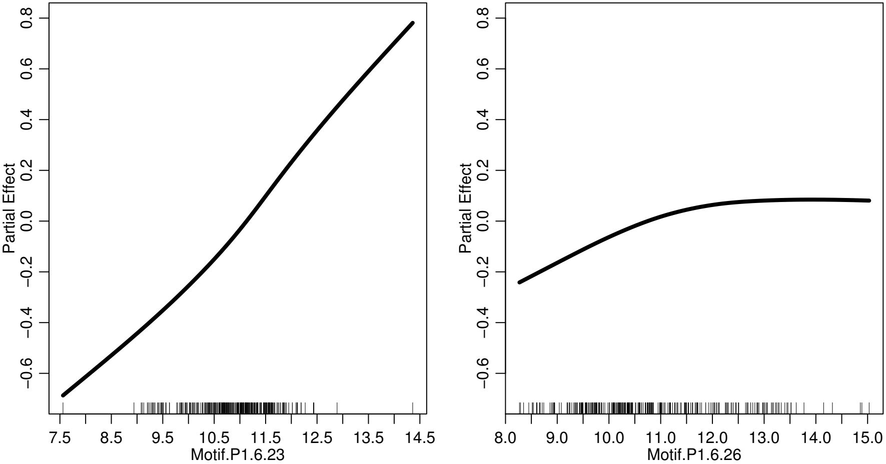


dotted line: $\lambda_2 = 0$

$\rightsquigarrow$ λ_2 seems not so important: just consider a few candidate values

(solid and dashed line)

motif regression: $n = 287, \ p = 195$



$\leadsto$ a linear model would be "fine as well"

Conclusions

if the problem is sparse and smooth:
only a few $X^{(j)}$'s influence Y (only a few non-zero f_j^0) and the non-zero f_j^0 are smooth
$\leadsto$ one can often afford to model and fit additive functions in high dimensions

reason:

- dimensionality is of order $\mathrm{dim} = O(pn)$
  $\log(\mathrm{dim})/n = O((\log(p) + \log(n))/n)$ which is still small
- sparsity and smoothness then lead to: if each f_j^0 is twice continuously differentiable

$$\|\hat{f} - f^0\|_2^2/n = O_P(\underbrace{\text{sparsity}}_{\text{no. of non-zero } f_j^0} \sqrt{\log(p)}n^{-4/5})$$

(cf. Ch. 8.4 in Bühlmann & van de Geer (2011))