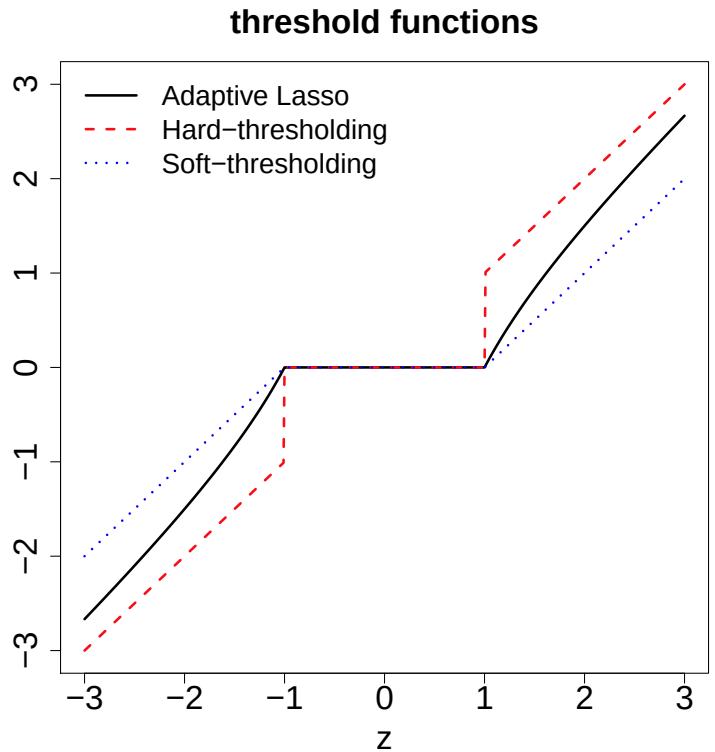
Adaptive Lasso

is a good way to address the bias problems of the Lasso

for orthonormal design



two-stage procedure:

- initial estimator $\hat{\beta}_{\text{init}}$, e.g., the Lasso
- re-weighted ℓ_1-penalty

$$\hat{\beta}_{\text{adapt}}(\lambda) = \text{argmin}_\beta \left(\|Y - X\beta\|_2^2/n + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{|\hat{\beta}_{\text{init},j}|} \right)$$

at least as sparse (typically more sparse) than Lasso

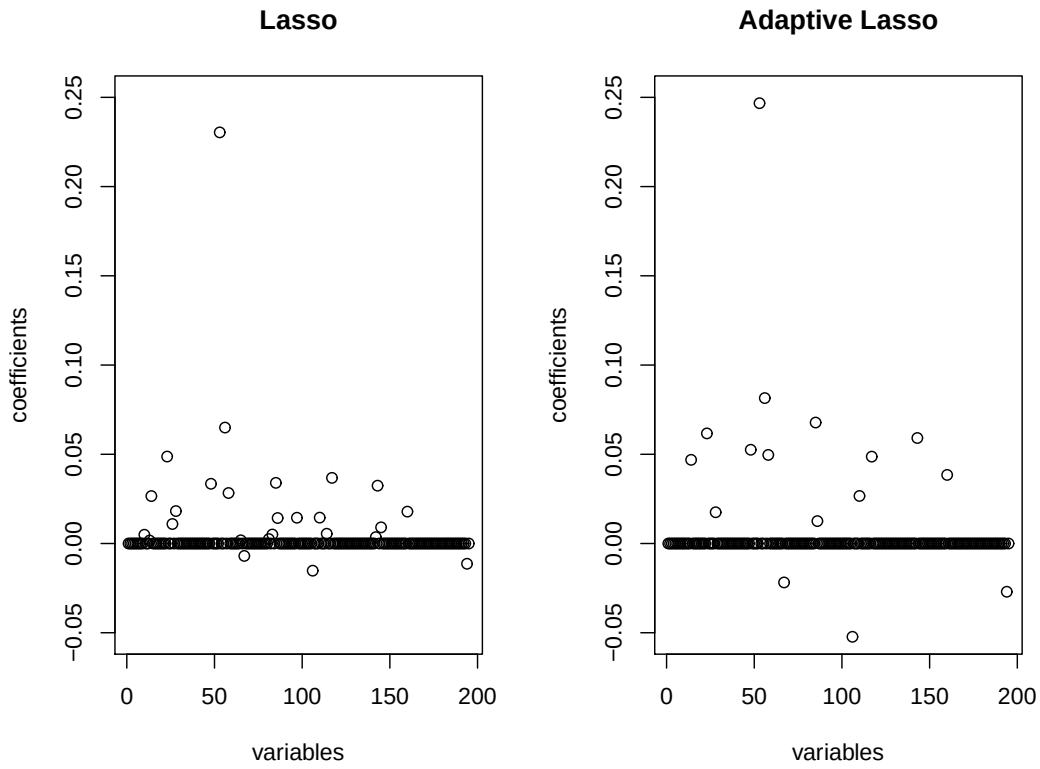
- "vaguely speaking":
  adaptive Lasso between ℓ_1- and ℓ_0-penalty methods

$$\hat{\beta}_{\ell_0}(\lambda) = \text{argmin}_\beta \left(\|Y - X\beta\|_2^2/n + \lambda\|\beta\|_0 \right)$$

  e.g. AIC, BIC, etc.
- adaptive Lasso has better theoretical properties than Lasso for variable screening (and selection) if the truth is assumed to be sufficiently sparse

alternatives: thresholding the Lasso; Relaxed Lasso

The adaptive Lasso workhorse



$$p = 195, n = 143, |\hat{S}_{\text{ada}-\text{Lasso}}(\lambda_{CV})| = 16$$

we will discuss later in the course the issue of assigning "significance of selected variables"

should we always use the adaptive Lasso?

- it's slightly more complicated – need two Lasso fits
- I tend to say:
  "Yes, often the adaptive Lasso is perhaps a bit better"

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:

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

this implies (by setting the sub-differential to zero) 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!

coordinate descent algorithm 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:

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

"we probe the gradient when setting jth comp. to zero"

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 $\{1, \ldots, p\}$:
$\mathcal{S}^{[m]} = \mathcal{S}^{[m-1]} + 1 \bmod p$. Abbreviate by $j = \mathcal{S}^{[m]}$ the value of $\mathcal{S}^{[m]}$.
4: if $|G_j(\beta_{-j}^{[m-1]})| \leq \lambda$: set $\beta_j^{[m]} = 0$,
otherwise: $\beta_j^{[m]} = \text{argmin}_{\beta_j} Q_\lambda(\beta_{+j}^{[m-1]})$,
where $\beta_{-j}^{[m-1]}$ is the parameter vector where the jth component is set to zero and $\beta_{+j}^{[m-1]}$ 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`

in addition to the KKT solutions
(Lemma 2.1, Bühlmann and van de Geer (2011)):

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

if $|G_j(\hat{\beta})| = \lambda$ there is some "ambiguity": but $\hat{\beta}_j = 0$ and $|G_j(\hat{\beta})| = \lambda$ happens with probability zero if the compatibility condition holds
and the following is true:

if the solution of the Lasso optimization is not unique ($p > n$):

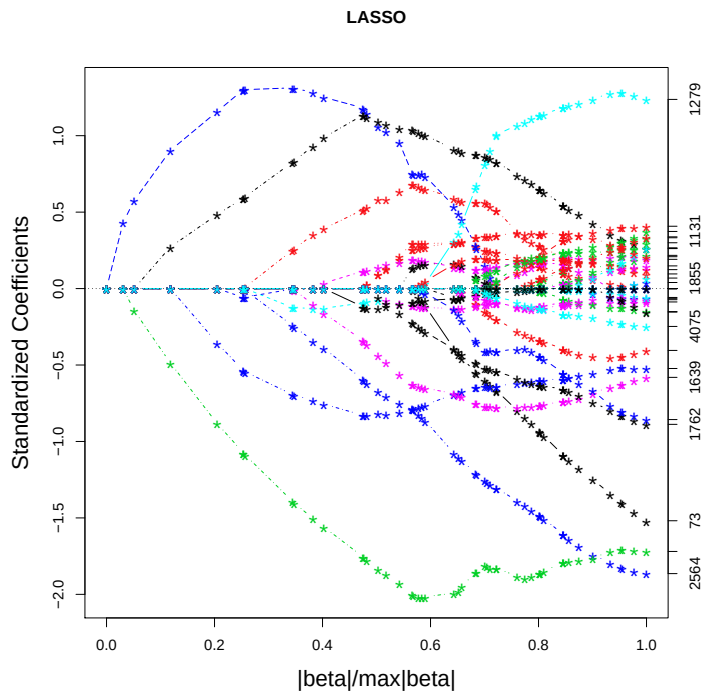
$$\text{if } G_j(\hat{\beta}) < \lambda \text{ for some solution } \hat{\beta}_j$$
$$\Longrightarrow \ \hat{\beta}_j = 0 \text{ for all solutions}$$

$\rightsquigarrow$ uniqueness of the estimated zeros !

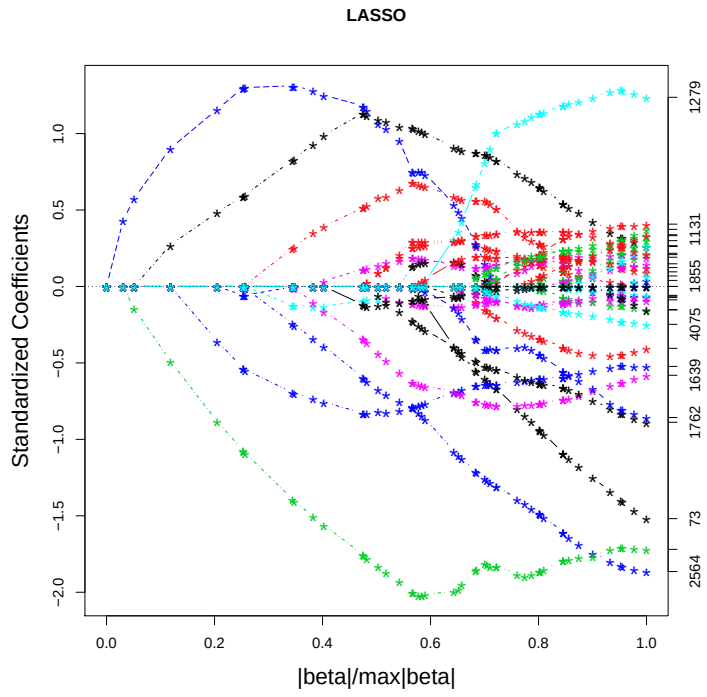
The Lasso regularization path

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 $\lambda_{\max} = \max_{j=1,\ldots,p} |2X_j^T Y/n|$: $\hat{\beta}(\lambda_{\max}) = 0$
  (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.

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

III. Generalized linear models (GLMs)

(Ch. 3 in Bühlmann and van de Geer (2011))

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 $\pi_i = \mathbb{P}[Y_i = 1|X_i]$

$\log(\pi_i/(1-\pi_i)) = \mu + X_i^T\beta,\ \pi_i = \frac{\exp(\mu+X_i^T\beta)}{1+\exp(\mu+X_i^T\beta)}$

log-likelihood

$$\sum_{i=1}^{n} \log(\pi_i^{Y_i}(1-\pi_i)^{1-Y_i}) = \sum_{i=1}^{n}(Y_i \log(\pi_i) + (1-Y_i)\log(1-\pi_i))$$

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

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:

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

(typically) unpenalized intercept

note: often used nowadays for classification with deep neural networks

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

estimator:

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

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

if somebody gives you the feature mapping $\phi_\theta(\cdot)$ (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
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

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

seven positions around "GT"

training data:

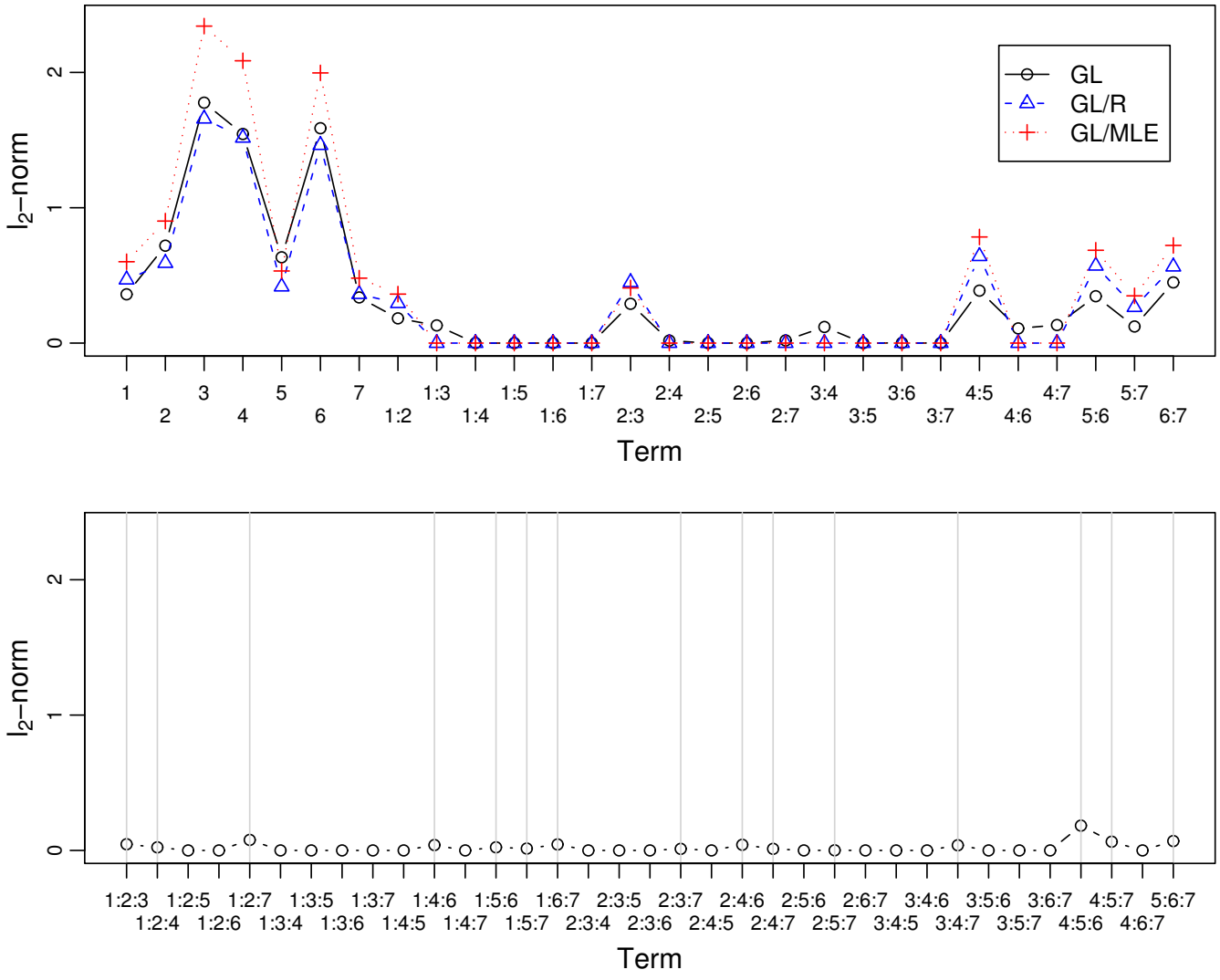
$$Y_i \in \{0, 1\} \text{ true donor site or not}$$
$$X_i \in \{A, C, G, T\}^7 \text{ positions}$$
$$i = 1, \ldots, n \approx 188'000$$

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)
$\rightsquigarrow$ 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

block coordinate descent (updates on blocks of coefficients)

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