Recap

$$Y = X\beta^0 + \varepsilon, \ \ p \gg n$$

for the Lasso:
Theorem 6.1 in Bühlmann and van de Geer (2011)
assume: compatibility condition holds with compatibility constant $\phi_0^2 \geq L > 0$
Then, on $\mathcal{T}$ and for $\lambda \geq 2\lambda_0$:

$$\|X(\hat{\beta} - \beta^0\|_2^2/n + \lambda\|\hat{\beta} - \beta^0\|_1 \leq 4\lambda^2 s_0/\phi_0^2$$

When does the compatibility condition hold?

Corollary 6.8 from Bühlmann and van de Geer (2011) – modified form

Assume that the row vectors of X are i.i.d. sampled from a sub-Gaussian distribution with mean zero and covariance matrix Σ. Assume that

- $\lambda^2_{min}(\Sigma) > 0$
- $s_0 = |S_0| = O(\sqrt{n/\log(p)})$

Then: $\phi_0^2 \geq \lambda^2_{\min}(\Sigma) > 0$ with probability $\to 1$ ($n \to \infty$)

Example: Toeplitz matrix $\Sigma_{ij} = \rho^{|i-j|}$ $(0 \leq \rho < 1)$:
$\lambda^2_{min}(\Sigma) \geq L > 0$ where L is independent of p

Variable screening

active set (of variables): $S_0 = \{j;\ \beta_j^0 \neq 0\}$
estimated active set: $\hat{S}_0 = \{j;\ \hat{\beta}_j \neq 0\}$

make an assumption that true regression coefficients are not too small

"beta-min condition" : $\min_{j \in S_0} |\beta_j^0| > \underbrace{4\lambda s_0/\phi_0^2}_{\text{bound for } \|\hat{\beta} - \beta^0\|_1}$

$$\Longrightarrow \mathbb{P}[\hat{S} \supseteq S_0] \geq \mathbb{P}[\mathcal{T}] = \text{“large”}$$

Theory versus Practice

theory:

$$\mathbb{P}[\hat{S} \supseteq S_0] \to 1$$

if the following hold:

- compatibility condition for the (fixed) design X
- beta-min condition
- i.i.d. Gaussian errors (can be relaxed)

in addition: $|\hat{S}| \leq \min(n, p)$
hence: huge dimensionality reduction if $p \gg n$

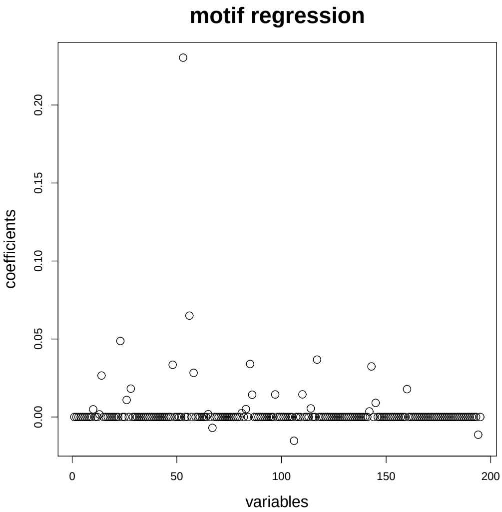
in practice: $\mathbb{P}[\hat{S} \supseteq S_0]$ may not be so large...
even if one chooses λ very small which results in a typically larger set $\hat{S}$...

possible reasons to explain with theory:

- compatibility constant ϕ_0^2 might be very small (due to highly correlated columns in X or near linear dependence among a few columns of X)
  $\leadsto \|\hat{\beta} - \beta^0\|_1 \le 4\lambda s_0/\phi_0^2$
  $\leadsto$ requires a stronger beta-min condition!
- errors are non-Gaussian (heavy tailed)

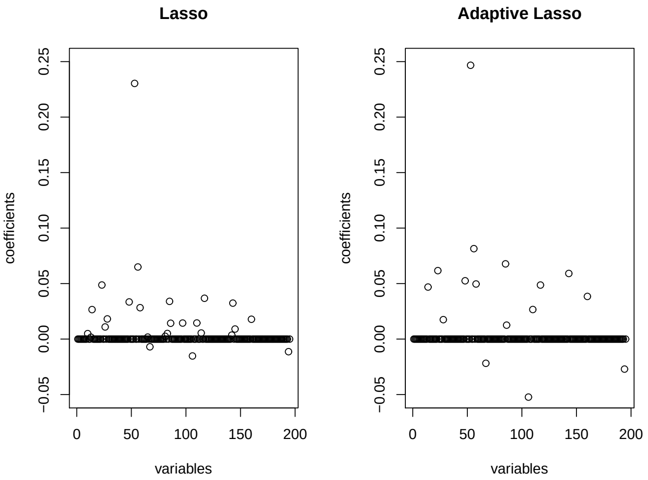
it is "empirically evident" though: $\mathbb{P}[\hat{S} \supseteq S_{\mathrm{substantial}(C)}]$ large
where $S_{\mathrm{substantial}(C)} = \{j;\ |\beta_j^0| \ge \underbrace{C}_{\text{large}}\}$

The Lasso workhorse



$$p = 195, n = 143, |\hat{S}(\lambda_{CV})| = 26$$

The (adaptive) Lasso workhorse



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

Variable selection

under more restrictive irrepresentable condition or neighborhood stability condition on the design X and assuming beta-min condition $\min_{j \in S_0} |\beta_j^0| \gg \sqrt{s_0 \log(p)/n}$:

$$\mathbb{P}[\hat{S} = S_0] \to 1 \ (n \to \infty)$$

the irrepresentable condition is sufficient and essentially necessary for consistent variable selection

this condition is often not fulfilled in practice
(and choosing the correct λ would be difficult as well)

$\leadsto$ variable screening is realistic ("choose λ by CV")
variable selection is not very realistic

better "translation":
LASSO = Least Absolute Shrinkage and Screening Operator

Variable selection

under more restrictive irrepresentable condition or
neighborhood stability condition on the design X
and assuming beta-min condition $\min_{j \in S_0} |\beta_j^0| \gg \sqrt{s_0 \log(p)/n}$:

$$\mathbb{P}[\hat{S} = S_0] \to 1 \ (n \to \infty)$$

the irrepresentable condition is sufficient and essentially necessary for consistent variable selection

this condition is often not fulfilled in practice
(and choosing the correct λ would be difficult as well)

$\leadsto$ variable screening is realistic ("choose λ by CV")
variable selection is not very realistic
better "translation":
LASSO = Least Absolute Shrinkage and Screening Operator

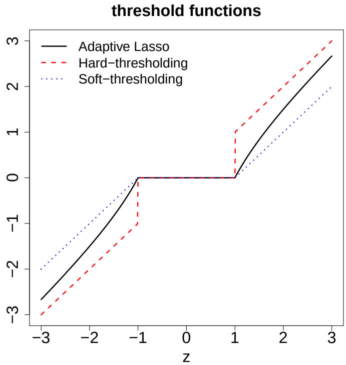
version of Table 2.2 in the book:

property	design condition	size of non-zero coeff.
slow prediction conv. rate	no requirement	no requirement
fast prediction conv. rate	compatibility	no requirement
estimation error bound $\|\hat{\beta} - \beta^0\|_1$	compatibility	no requirement
variable screening	compatibility or restricted eigenvalue	beta-min condition weaker beta-min cond.
variable selection	neighborhood stability $\Leftrightarrow$ irrepresentable cond.	beta-min condition

Adaptive Lasso

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

for orthonormal design



two-stage procedure:

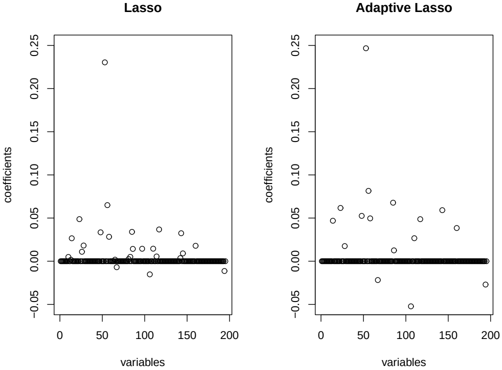
- initial estimator $\hat{\beta}_{\mathrm{init}}$, e.g., the Lasso
- re-weighted ℓ_1-penalty

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

adaptive Lasso often works well in practice (more sparse than Lasso) and has better theoretical properties than Lasso for variable screening (and selection) if the truth is assumed to be sparse

alternatives: thresholding the Lasso; Relaxed Lasso

The adaptive Lasso workhorse



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

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
- the differences in large-scale data are perhaps not so large
- 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:

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

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!

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 $\{1, \ldots, p\}$:
$\mathcal{S}^{[m]} = \mathcal{S}^{[m-1]} + 1 \bmod p$. 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)

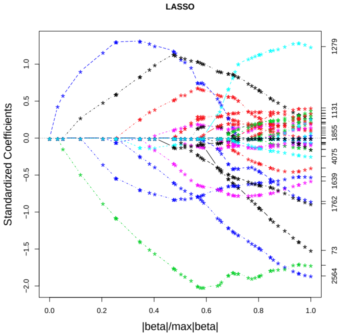
active set strategy can speed up the algorithm for sparse cases: mainly work on the non-zero coordinates and up-date all coordinates e.g. every 20th times

`R`-package `glmnet`

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 $\lambda_{\max} = |2X^T Y/n|$: $\hat{\beta}(\lambda_{\max}) = 0$
  (because of KKT conditions!)



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

Generalized linear models (GLMs)

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

$$\text{GLM:} \quad Y_1, \ldots, Y_n \text{ independent}$$

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

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

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

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

which is a convex function in μ, β

Lasso for linear logistic regression:

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

note: often used nowadays for classification with deep neural networks

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

estimator:

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

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

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

Group Lasso

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

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

seven positions around "GT"

training data:

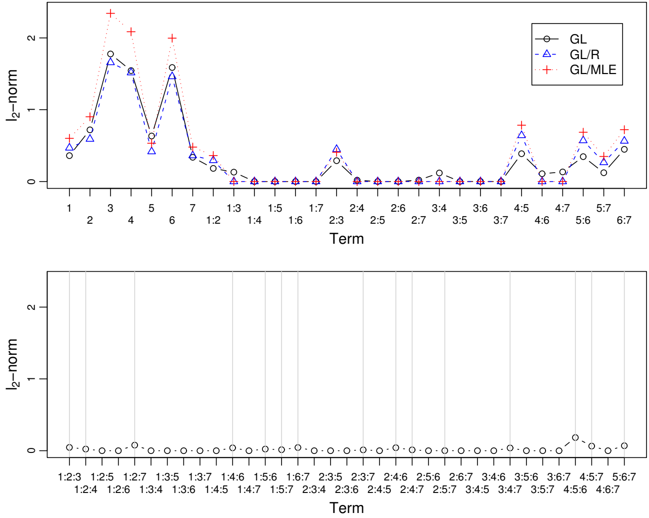
$$\begin{aligned} &Y_i \in \{0,1\} \text{ true donor site or not} \\ &X_i \in \{A, C, G, T\}^7 \text{ positions} \\ &i = 1, \ldots, \approx 188'000 \end{aligned}$$

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

block coordinate descent

Algorithm 1 Block Coordinate Descent Algorithm

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

repeat

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 $\mathscr{S}^{[m]}$.

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).

until numerical convergence