Topic 1: Linear regression and shrinkage estimation
STA7233, Advanced Regression, Fall 2022
Last updated on September 14, 2022

Disclaimer: These notes may contain errors. These and other course materials are not be shared or distributed without explicit consent from the instructor.

Contents

1 Ordinary least squares

We will begin this section by describing the classical linear regression model. Suppose we have $(x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^p$ is the p-dimensional set of measured predictors for the ith subject in the study. Throughout this section, we treat the x_i as fixed. The classical linear regression model assumes that $(y_1, \ldots, y_n)$, the measured responses, are realizations of the random vectors

$$Y_i = \beta_0^* + \sum_{j=1}^{p} x_{ij}\beta_j^* + \epsilon_i, \quad , i = 1, \ldots, n,$$

where $\mathrm{E}(\epsilon_i) = 0$, $\mathrm{Var}(\epsilon_i) = \sigma^2$, ϵ_i is independent of ϵ_j for all $i \neq j$, $\beta_0^* \in \mathbb{R}$ is the unknown intercept, and $\beta^* = (\beta_1^*, \ldots, \beta*_p)' \in \mathbb{R}^p$ is the unknown regression coefficient vector. The superscript $*$ is used to distinguish population parameters from optimization variables.

The standard estimator of β_0^* and β^* is the *ordinary least squares* (OLS) estimator. This estimator is the value of $(\beta_0, \beta) \in \mathbb{R}^{p+1}$ which minimizes the residual sum of squares criterion, i.e., the least squares estimator is

$$\underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2.$$

To derive the least squares estimator, we change notation slightly. Let $\boldsymbol{X} \in \mathbb{R}^{n \times p+1}$ be a matrix whose ith row is $(1, x_{i1}, \ldots, x_{1p})$. Similarly, let $\boldsymbol{y} = (y_1, \ldots, y_n)' \in \mathbb{R}^n$ so that we can write $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)' \in \mathbb{R}^{p+1}$ and

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 = \boldsymbol{y}'\boldsymbol{y} - 2\boldsymbol{y}'\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}.$$

Since the residual sum of squares criterion is convex in $\boldsymbol{\beta}$, we know that the least squares estimator, $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$, is the value of $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ which satisfies the *first order optimality condition*:

$$\nabla f(\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}) = 0.$$

Thus, using that $\nabla f(\boldsymbol{\beta}) = -2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}$ it follows that

$$\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \implies \nabla f(\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}) = 0,$$

and therefore, the least squares estimator is given by

$$\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y},$$

assuming $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ exists.

There are a number of well-known facts about $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$ that will be left as review exercises.

Exercise. If $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$ for $i = 1, \ldots, n$, show that

$$\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} \sim \mathrm{N}_{p+1}\left(\boldsymbol{\beta}^*, \sigma^2(\boldsymbol{X}'\boldsymbol{X})^{-1}\right).$$

Exercise. Letting $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} \in \mathbb{R}^n$ be the fitted values, and letting

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{\boldsymbol{y}})^2,$$

prove that (i) $\hat{\sigma}^2$ is unbiased for σ^2 and (ii)

$$\frac{(n-p-1)\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1}.$$

A well-known result in statistics about the least squares estimator is the Gauss-Markov theorem.

Gauss-Markov theorem

Suppose that $Y_i = \beta_0^* + \sum_{j=1}^{p} x'_{ij}\beta_j^* + \epsilon_i, \quad , i = 1, \ldots, n$. If the ϵ_i have mean zero and finite variance σ^2 for $i = 1, \ldots, n$, and the ϵ_i are uncorrelated, $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$ satisfies:

$$\mathrm{Var}(\tilde{\boldsymbol{\beta}}) \succeq \mathrm{Var}(\hat{\boldsymbol{\beta}}^{\mathrm{OLS}})$$

for all estimators $\tilde{\boldsymbol{\beta}}$ which are linear functions of $\boldsymbol{y}$ and unbiased, where the notation $A \succeq B$ means $A - B$ is non-negative definite. That is, $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$ has the minimum variance amongst all linear unbiased estimators, or restated, $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$ is the best linear unbiased estimator (BLUE).

However, there is one major drawback of the Gauss-Markov theorem: restricting ourselves only to unbiased estimators does not allow us to exploit the so-called *bias-variance tradeoff*.

2 Bias-variance tradeoff

The bias-variance tradeoff refers to a phenomenon where estimators with lower bias tend to have higher variance and vice versa. This can be illustrated mathematically via the

decomposition of mean squared error into variance and bias squared:

$$\mathrm{MSE}(\hat{\beta}, \beta^*) := \mathrm{E}(\|\hat{\beta} - \beta^*\|_2^2) = \underbrace{\mathrm{E}\{\|\hat{\beta} - \mathrm{E}(\hat{\beta})\|_2^2\}}_{\text{Variance}} + \underbrace{\|\mathrm{E}(\hat{\beta}) - \beta^*\|_2^2}_{\text{Bias}^2}.$$

This decomposition suggests that in order to achieve the lowest possible mean-squared error, one may allow for a small amount of bias if this leads to a large reduction in variance. This notion is depicted in the illustration below. In the right panel, we see that estimators with near-zero bias often have substantially larger variance, and thus, larger mean squared error than do estimators with small, nonzero bias.

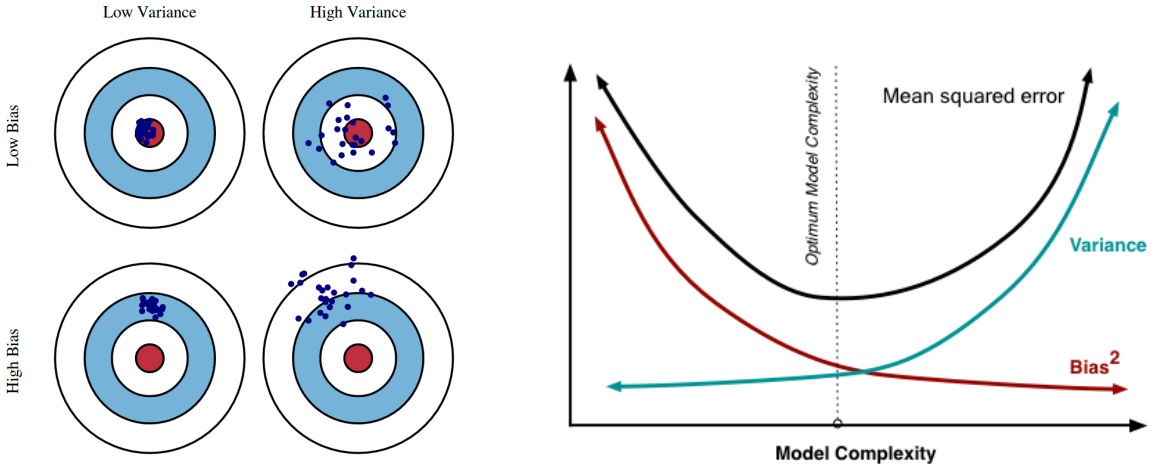

Figure 1: Illustrations of the bias-variance tradeoff. Taken from the blog of Scott Fortmann-Roe.

In light of this tradeoff, modern regression techniques often introduce a small amount of bias using *shrinkage estimation*, or more generally, *regularization*, in order to reduce variance. The first and perhaps more well-known of these estimators is the so-called *ridge regression* estimator.

3 Ridge regression

3.1 Definition

The (constrained) ridge regression estimator is

$$(\hat{\beta}_{0,\lambda}^{\text{Ridge}}, \hat{\beta}_{\lambda}^{\text{Ridge}}) = \underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}' \beta_j)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq t$$

where $t > 0$ is a user-specified tuning parameter. It is far more common to see this estimator written in its (equivalent) *Lagrangian* form

$$\underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}' \beta_j)^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2 \right\} \tag{1}$$

where $\lambda > 0$ is a user-specified tuning parameter. We will discuss methods for selecting λ later, so at this point, simply treat λ as fixed. Note that there is a one-to-one correspondence between λ and t, i.e., for any $t > 0$ for which the constraint is active (meaning the minimum occurs at the boundary of the set of β where $\sum_{j=1}^{p} \beta_j^2 = t$), there is a unique λ corresponding to that solution. However, determining t from λ, and vice versa, is non-trivial.

The ridge regression estimator is the first we will consider wherein we use shrinkage. Specifically, the ridge regression estimator can be described as a *penalized least squares* estimator as it minimizes a least squares criterion plus a *penalty* on the magnitude of the β_j's.

To compute the ridge regression estimator, we introduce new notation. First, let $\|a\|_2 = \sqrt{a'a}$ denote the euclidean norm of a vector a, let $\mathbf{y} = (y_1 - \bar{y}, \ldots, y_n - \bar{y})' \in \mathbb{R}^n$ and let $\mathbf{X} = (x_1 - \bar{x}, \ldots, x_n - \bar{x})' \in \mathbb{R}^{n \times p}$ where $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \in \mathbb{R}$ and $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \in \mathbb{R}^p$. We refer to $\mathbf{X}$ and $\mathbf{y}$ as the "column-centered" versions of $\boldsymbol{X}$ and $\boldsymbol{y}$. This way we can write the ridge regression estimator more succinctly since

$$\hat{\beta}_\lambda^{\text{Ridge}} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right\}$$

and $\hat{\beta}_{0,\lambda} = \bar{y} - \bar{x}' \hat{\beta}_\lambda$ are the minimizers of (1).

Exercise. Prove the previous statement.

First, we want to derive a closed form for the ridge regression estimator. Since the least squares criterion is convex in β and since $\|\beta\|_2^2 = \beta'\beta$ is convex, we know that any β which satisfies the first order optimality conditions (KKT conditions) is a minimizer. Notice,

$$\begin{aligned} f_\lambda(\beta) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 &= \frac{1}{2} \mathbf{y}'\mathbf{y} - \mathbf{y}'\mathbf{X}\beta + \frac{1}{2} \beta'\mathbf{X}'\mathbf{X}\beta + \frac{\lambda}{2} \beta'\beta \\ \implies \nabla f_\lambda(\beta) &= -\mathbf{X}'\mathbf{y} + \mathbf{X}'\mathbf{X}\beta + \lambda\beta \\ &= -\mathbf{X}'\mathbf{y} + (\mathbf{X}'\mathbf{X} + \lambda I_p)\beta \\ \implies \nabla f_\lambda(\beta) = 0 \iff \beta &= (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \mathbf{X}'\mathbf{y} \end{aligned}$$

so that we have

$$\hat{\beta}_\lambda^{\text{Ridge}} = (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \mathbf{X}'\mathbf{y}.$$

It is quick to check that $\hat{\beta}_\lambda^{\text{Ridge}} = (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{y}$ is a biased estimate of β^*. Let $\epsilon = (\epsilon_1, \ldots, \epsilon_n) \in \mathbb{R}^n$ be a mean zero vector of errors.

$$\begin{aligned}
\mathrm{E}(\hat{\beta}_\lambda^{\text{Ridge}}) &= \mathrm{E}\left[(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{y}\right] \\
&= \mathrm{E}\left[(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'(\mathbf{X}\beta^* + \epsilon)\right] \\
&= (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{X}\beta^* \neq \beta^*
\end{aligned}$$

Since as $\lambda \to 0$, $(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \approx (\mathbf{X}'\mathbf{X})^{-1}$, it is clear that ridge regression has least squares as an "edge-case" (i.e., letting the tuning parameter tend to one of its boundary points). Conversely, as $\lambda \to \infty$,

$$(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \approx \lambda^{-1} I_p,$$

so we see that $\hat{\beta}_\lambda^{\text{Ridge}} \to 0_p$, the origin in $\mathbb{R}^p$.

This feature be illustrated by examining a so-called "trace-plot" in Figure 2. This is a plot which shows the estimates of β along a path of tuning parameter values. In this example, and in those going forward, we will be using the "mice" data available from the R package `spls`. In this example, we predict the expression of a gene (at transcript 1415889_a_at) based on 10 of 145 microsatellite markers collected from the liver tissue of mice.

We see above that as λ gets larger, all coefficient estimates tend toward zero.

The R code for obtaining this plot is given by:

```
library(glmnet)
library(spls)
data(mice)
fit = glmnet(x = mice$x[,1:10], y = mice$y[,1], alpha = 0)
plot(fit, xvar="lambda")
```

3.2 Comparison to OLS

We now consider the variance of the ridge regression estimator relative to the OLS estimator. First, we have

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}_\lambda^{\text{Ridge}}) &= \mathrm{Var}\left[(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{y}\right] \\
&= \mathrm{Var}\left[(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'(\mathbf{X}\beta^* + \epsilon)\right] \\
&= \mathrm{Var}\left[(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\epsilon\right] \\
&= (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathrm{Var}(\epsilon)\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \\
&= \sigma^2(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}
\end{aligned}$$

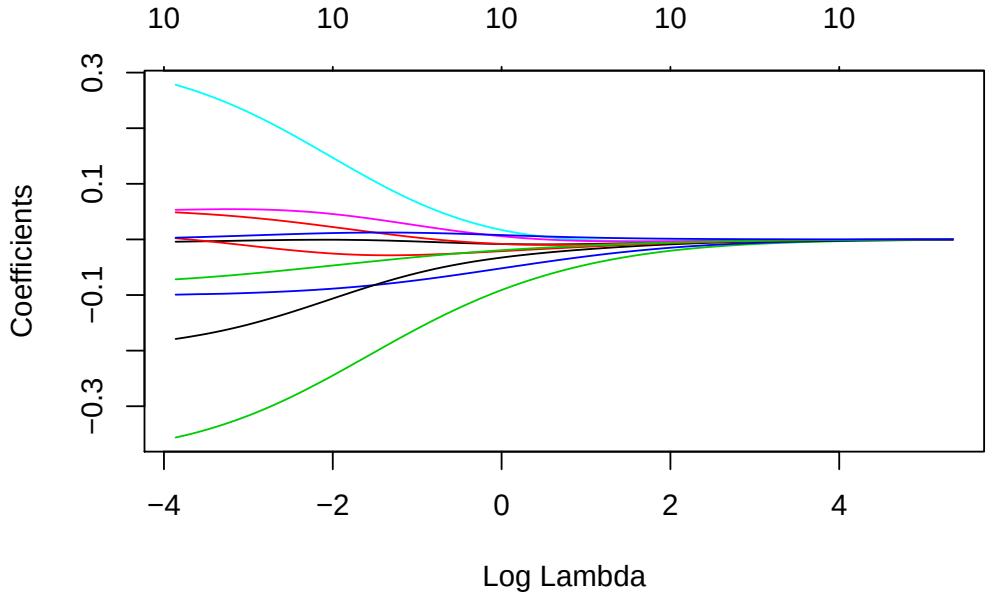

Figure 2: The output of plotting a `glmnet` fit with `alpha = 0`. This is a "solution path" or "trace plot", which displays the values of the estimated coefficients as a function of λ.

Recall that we know the OLS estimator has variance $\text{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$, so we see

$$\text{Var}(\hat{\beta}) - \text{Var}(\hat{\beta}_\lambda^{\text{Ridge}}) = \sigma^2\{(\mathbf{X}'\mathbf{X})^{-1} - (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\}.$$

We need only prove that this difference is non-negative definite, in which case we will have verified that the ridge regression estimator has variance smaller than the OLS estimator.

Let $UDU' = \mathbf{X}'\mathbf{X}$ be the eigendecomposition of $\mathbf{X}'\mathbf{X}$ and let $D_\lambda = D + \lambda I_p$. Then,

$$\begin{aligned}
\text{Var}(\hat{\beta}) - \text{Var}(\hat{\beta}_\lambda^{\text{Ridge}}) &= \sigma^2\{(\mathbf{X}'\mathbf{X})^{-1} - (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\} \\
&= \sigma^2(UD^{-1}U' - UD_\lambda^{-1}U'UDU'UD_\lambda^{-1}U') \\
&= \sigma^2 U(D^{-1} - D_\lambda^{-1}DD_\lambda^{-1})U'
\end{aligned}$$

so $\text{Var}(\hat{\beta}) - \text{Var}(\hat{\beta}_\lambda^{\text{Ridge}})$ has jth eigenvalue

$$\sigma^2\left(d_j^{-1} - \frac{d_j}{(d_j + \lambda)^2}\right) > 0$$

since all $d_j > 0$ and $\lambda > 0$. This verifies that the ridge regression estimator *always* has smaller variance than the OLS estimator. Of course, to have smaller MSE, it is a matter of

selecting λ to balance the variance reduction and bias.

MSE of ridge regression

There always exists a $\lambda > 0$ such that $\text{MSE}(\hat{\beta}_\lambda^{\text{Ridge}}) < \text{MSE}(\hat{\beta})$.

The proof of the previous fact will be given as a homework problem.

3.3 Computing the ridge regression estimator

Going forward, you should adhere to the *first rule of statistical computing*[1]:

First rule of statistical computing

Avoid matrix inversions whenever possible.

Instead of computing matrix inverses like $(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}$, our goal will always be to use matrix decompositions or iterative solvers. To show how to easily compute the ridge regression estimator for any λ, let

$$UDV' = \mathbf{X}$$

be the singular value decomposition of $\mathbf{X}$. Then, since

$$\mathbf{X}'\mathbf{X} = VDU'UDV' = VD^2V'$$

it follows that

$$\begin{aligned} \mathbf{X}'\mathbf{X} + \lambda I_p &= VD^2V' + V(\lambda I_p)V' \\ &= V(D^2 + \lambda I_p)V' \\ \implies (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} &= V(D^2 + \lambda I_p)^{-1}V' \end{aligned}$$

and thus

$$\begin{aligned} (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}'\mathbf{y} &= V(D^2 + \lambda I_p)^{-1}V'VDU'\mathbf{y} \\ &= V(D^2 + \lambda I_p)^{-1}DU'\mathbf{y} \\ &= V\tilde{D}U'\mathbf{y} \end{aligned}$$

where $\tilde{D}$ is a diagonal matrix with jth diagonal element $d_j/(d_j^2 + \lambda)$ where d_j is the jth singular value of $\mathbf{X}$.

[1] Inspired by Hua Zhou's statistcal computing lecture notes

3.4 Cross-validation

There are a number of methods for selecting the tuning parameter $\lambda > 0$, but without question, the most widely used is cross-validation. Traditionally, we use what is known as K-fold cross-validation (e.g., leave-one-out cross-validation is simply n-fold cross-validation). The idea is relatively simple, but we first define two important concepts that will be used throughout the semester.

- **Training set**: the data we use to actually fit the model, i.e., the $\mathbf{X}$ and $\mathbf{Y}$ we plug into the ridge regression criterion.
- **Testing set**: the data we use to validate the model, i.e., data not used in fitting the model.

The idea of K-fold cross-validation is to take our original n observations, split them in K "folds". Then, for each of the K-folds, we fit the model to the data outside the Kth fold (training set) and measure model fit on the Kth fold (testing set). Specifically, let $C_1, \ldots, C_K$ be a K element partition of $\{1, \ldots, n\}$ where each C_i is of (approximately) the same cardinality and membership to each C_i is assigned completely at random.

Before formally stating the cross-validation criterion, we need to perform some important house cleaning. Define $X_{-C_i} \in \mathbb{R}^{n-|C_i| \times p}$ and $y_{-C_i} \in \mathbb{R}^{n-|C_i|}$ as the submatrices and subvector of X and y, respectively with the rows and elements indexed by the set C_i removed. *Note that these are **not** the versions of the data which have columnwise mean zero from above*. Then, we define the following:

- $\mathbb{X}_{-C_i} = X_{-C_i} - 1_{n-|C_i|}\bar{X}'_{-C_i}$ where $\bar{X}_{-C_i} = \frac{1}{n-|C_i|}\sum_{j \notin C_i} x_{j,\cdot}$
- $\tilde{y}_{-C_i} = y_{-C_i} - \bar{y}'_{-C_i}$ where $\bar{y}_{-C_i} = \frac{1}{n-|C_i|}\sum_{j \notin C_i} y_j$
- $\mathbb{X}_{C_i} = X_{C_i} - 1_{|C_i|}\bar{X}'_{-C_i}$
- $\tilde{y}_{C_i} = y_{C_i} - \bar{y}'_{-C_i}$

Here, $\mathbb{X}_{-C_i}, \tilde{y}_{-C_i}$ are the "training data" and $\mathbb{X}_{C_i}, \tilde{y}_{C_i}$ are the "testing data". By subtracting off training set means from both sets, we can again ignore the intercept.

Letting $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_s\}$ denote the set of candidate tuning parameters, and letting

$$\tilde{\beta}^{\text{Ridge}}_{-C_i,\lambda} = \arg\min_{\beta} \left\{ \frac{1}{2(n-|C_i|)} \|\tilde{y}_{-C_i} - \mathbb{X}_{-C_i}\beta\|_2^2 + \frac{\lambda}{2}\|\beta\|_2^2 \right\},$$

(noting that we now divide the least squares criterion by the sample size so that the sum of squares has approximately the same order of magnitude in both cross-validation and the complete dataset) the cross-valdiation criterion selects the tuning parameter λ so that

$$\hat{\lambda} \in \arg\min_{\lambda \in \boldsymbol{\lambda}} \sum_{k=1}^{K} \|\tilde{y}_{C_i} - \mathbb{X}_{C_i}\hat{\beta}^{\text{Ridge}}_{-C_i,\lambda}\|_2^2.$$

Thus, in practice, our chosen estimator would be

$$\tilde{\beta}^{\text{Ridge}}_{\hat{\lambda}} = \underset{\beta \in \mathbb{R}}{\arg\min} \left\{ \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\hat{\lambda}}{2} \|\beta\|_2^2 \right\}.$$

Note again that this is equivalent to our original ridge regression estimator with tuning parameter scaled by n. Fortunately, many existing software packages in R do this behind the scenes.

```
set.seed(7934)
library(glmnet)
n <- 250
p <- 500
SigmaX <- matrix(.9, nrow=p, ncol=p)
diag(SigmaX) <- 1
eo <- eigen(SigmaX)
SigmaXsqrt <- tcrossprod(tcrossprod(eo$vec, diag(eo$val^.5)), eo$vec)
X <- matrix(rnorm(p*n), nrow=n)%*%SigmaXsqrt
B <- rep(.1, p)*sample(c(-1, 1), p, replace=TRUE)
y <- X%*%B + rnorm(n, sd = 2)
lambda.candidates <- 10^seq(4, -4, length=50)
cv.ridgefit <- cv.glmnet(x = X, y = y, lambda = lambda.candidates, alpha = 0)
plot(cv.ridgefit)
str(cv.ridgefit)
```

The output then looks like:

```
List of 11
 $ lambda    : num [1:50] 10000 6866 4715 3237 2223 ...
 $ cvm       : num [1:50] 5.94 5.85 5.71 5.55 5.38 ...
 $ cvsd      : num [1:50] 0.479 0.471 0.46 0.446 0.431 ...
 $ cvup      : num [1:50] 6.42 6.32 6.17 6 5.81 ...
 $ cvlo      : num [1:50] 5.46 5.38 5.25 5.11 4.95 ...
 $ nzero     : Named int [1:50] 500 500 500 500 500 500 500 500 500 500 ...
  ..- attr(*, "names")= chr [1:50] "s0" "s1" "s2" "s3" ...
 $ call      : language cv.glmnet(x = X, y = y, lambda = lambda.candidates, alpha = 0)
 $ name      : Named chr "Mean-Squared Error"
  ..- attr(*, "names")= chr "mse"
 $ glmnet.fit:List of 12
  ..$ a0       : Named num [1:50] 0.18 0.182 0.184 0.188 0.192 ...
  .. ..- attr(*, "names")= chr [1:50] "s0" "s1" "s2" "s3" ...
  ..$ beta     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  .. .. ..@ i       : int [1:25000] 0 1 2 3 4 5 6 7 8 9 ...
```

```
  .. .. ..@ p       : int [1:51] 0 500 1000 1500 2000 2500 3000 3500 4000 4500 ...
  .. .. ..@ Dim     : int [1:2] 500 50
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : chr [1:500] "V1" "V2" "V3" "V4" ...
  .. .. .. ..$ : chr [1:50] "s0" "s1" "s2" "s3" ...
  .. .. ..@ x       : num [1:25000] -0.000276 -0.000284 -0.000294 -0.000285 -0.000311 ..
  .. .. ..@ factors : list()
  ..$ df       : int [1:50] 500 500 500 500 500 500 500 500 500 500 ...
  ..$ dim      : int [1:2] 500 50
  ..$ lambda   : num [1:50] 10000 6866 4715 3237 2223 ...
  ..$ dev.ratio: num [1:50] 0.049 0.0648 0.0868 0.1126 0.1407 ...
  ..$ nulldev  : num 1555
  ..$ npasses  : int 645
  ..$ jerr     : int 0
  ..$ offset   : logi FALSE
  ..$ call     : language glmnet(x = X, y = y, lambda = lambda.candidates, alpha = 0)
  ..$ nobs     : int 250
  ..- attr(*, "class")= chr [1:2] "elnet" "glmnet"
 $ lambda.min: num 1.21
 $ lambda.1se: num 720
 - attr(*, "class")= chr "cv.glmnet"
```

You will notice in the output that there are two values of lambda highlighted: `lambda.min` and `lambda.1se`. `lambda.min` is the value I've defined above. The tuning parameter in `lambda.1se` is that chosen by applying the so-called "one-standard error rule".

This rule suggests that rather than selecting the tuning parameter yielding the minimum cross-validation error averaged over the folds, one would choose the largest tuning parameter whose averaged cross-validated error is less than the minimum cross-validation error plus one standard error (i.e., the standard deviation of the K cross-validated errors divided by square-root of K). This approach accounts for the randomness in the training/testing splits by selecting the most parsimonious model which is not significantly worse (in the statistical sense) than any other model.

A brief aside on standardization

One other aspect of ridge regression which is worth mentioning explicitly is that as written above, each β is penalized to the same degree. If the scales of the predictors are different, this can often lead to poor estimates. Thus, it is common to standardize predictors for model fitting so that each has columnwise standard deviation equal to one. Of course, care must be taken to perform the standardization appropriately when performing cross-validation. Specifically, we would use

- $\mathbb{X}_{-C_i} = (X_{-C_i} - 1_{n-|C_i|}\bar{X}'_{-C_i})/1_{n-|C_i|}\text{SD}(X_{-C_i})'$ where $\text{SD}(X_{-C_i}) \in \mathbb{R}^p$ are the columnwise standard deviations of X_{-C_i}

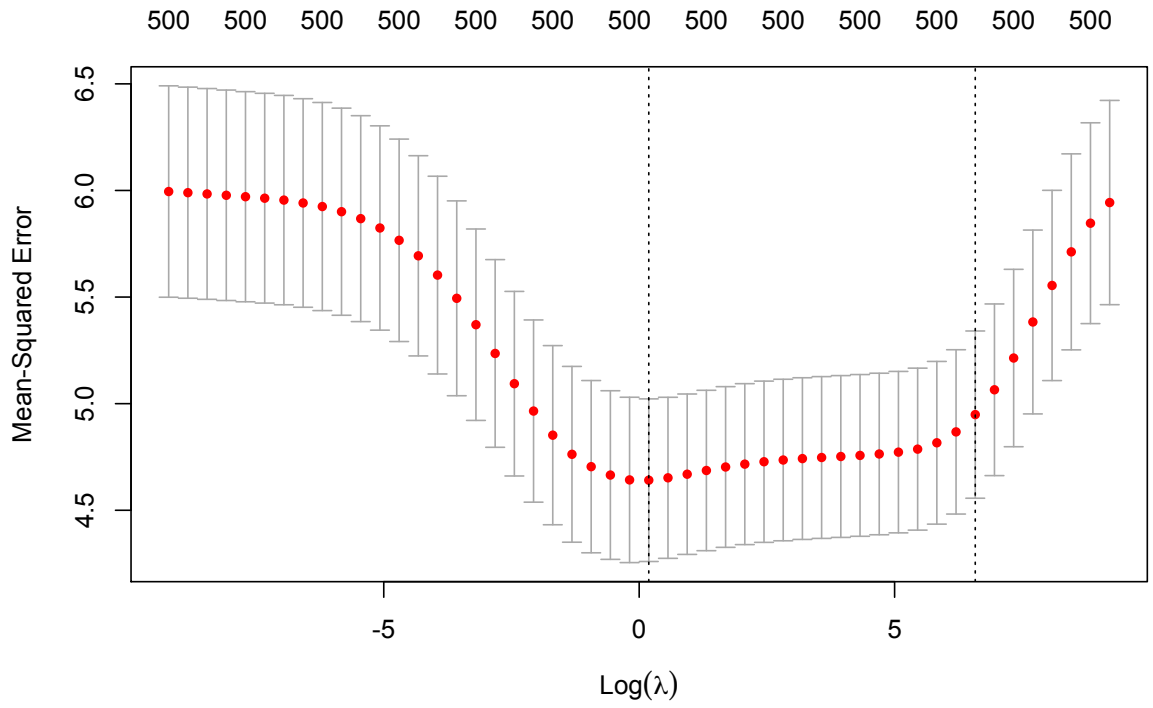

Figure 3: Cross-validation curve for ridge regression data.

- $\tilde{y}_{-C_i} = y_{-C_i} - \bar{y}'_{-C_i}$ where $\bar{y}_{-C_i} = \frac{1}{n-|C_i|} \sum_{j \notin C_i} y_j$
- $\mathbb{X}_{C_i} = (X_{C_i} - 1_{|C_i|} \bar{X}'_{-C_i}) / 1_{|C_i|} \text{SD}(X_{-C_i})'$
- $\tilde{y}_{C_i} = y_{C_i} - \bar{y}'_{-C_i}$

as training and testing datasets.

3.5 "High-dimensional" settings

In this section, we will address the question of when and why one would use ridge regression. In the previous section, we implicitly assumed that $n > p$: for example, when we defined the least squares estimator,

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y},$$

we did not discuss whether the inverse $(\mathbf{X}'\mathbf{X})^{-1}$ actually exists. Consider the case that it does not, e.g., if $n < p$. Then, it turns out that

$$\arg\min_{\beta \in \mathbb{R}^p} \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

is an infinite set, i.e., there are infinitely many OLS estimators. This is because, to satisfy the first order optimality conditions, we need

$$\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{X}'\mathbf{y}.$$

First, I will show how to construct one solution. Let $UDU' = \mathbf{X}'\mathbf{X}$ be the eigendecomposition of $\mathbf{X}'\mathbf{X}$. Now, define the function $f(a) = a^{-1}\mathbf{1}(a > 0)$ and let

$$(\mathbf{X}'\mathbf{X})^{-} \equiv Uf(D^{-1})U'$$

denote the Moore-Penrose pseudoinverse of $\mathbf{X}'\mathbf{X}$. It is quick to check that indeed, if

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{y},$$

we have

$$\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{X}'\mathbf{y},$$

so $\hat{\beta}$ is one least squares estimator. Now let us define a new estimator $\tilde{\beta}_C = C + \hat{\beta}$ where

$$C \in \mathbb{R}^p, \quad C \in \text{Null}(\mathbf{X}'\mathbf{X}) \setminus \{0\}$$

i.e., $(\mathbf{X}'\mathbf{X})C = 0$ where Null denotes the nullspace of a matrix. Since $\mathbf{X}'\mathbf{X}$ has at most rank n, we know by the rank-nullity theorem that its nullspace must have at least dimension $p - n$, so a nonzero C must exist. For example, take C to be any vector orthogonal to all of the leading $p - n - 1$ eigenvectors of $\mathbf{X}'\mathbf{X}$. Then, we can see that

$$\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{X}'\mathbf{y} \implies \mathbf{X}'\mathbf{X}C + \mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{X}'\mathbf{y} \implies \mathbf{X}'\mathbf{X}\tilde{\beta}_C = \mathbf{X}'\mathbf{y},$$

so $\tilde{\beta}_C$ is also a minimizer of the least squares criterion, as is $\tilde{\beta}_{aC}$ for any constant a.

At this point, you should see why ridge regression is so often used in high-dimensional settings. By adding a small constant to the diagonal of $\mathbf{X}'\mathbf{X}$, $\mathbf{X}'\mathbf{X} + \lambda I_p$, we are ensured that the ridge regression estimator is unique since $(\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1}$ always exists. Ridge regression is also commonly used in settings where $n > p$ and $\mathbf{X}$ has (approximately) linearly dependent columns. This is why ridge regression is often characterized as the remedy for collinearity.

4 Lasso

4.1 Definition

If $p > n$ and we use ridge regression, we may have a difficult time interpreting our model since all p predictors will have nonzero coefficient estimates. An alternative to ridge regression is the so-called *lasso* (**l**east **a**bsolute **s**hrinkage and **s**election **o**perator). This

estimator is similar to the ridge regression estimator, but rather than penalizing the sum of squared elements of β, it penalizes the sum of their absolute values:

$$\hat{\beta}_\lambda^{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

There are a number of important differences between the lasso and the ridge regression estimator:

- The lasso does not have a closed form solution in general: it often must be computed using some type of iterative algorithm.
- Like ridge regression, as $\lambda \to \infty$, $\hat{\beta}_\lambda^{\text{lasso}} \to 0$, but for intermediate values of λ, it is often the case that $\hat{\beta}_\lambda^{\text{lasso}}$ will have some elements *exactly* equal to zero.

The second point is the reason many refer to the lasso as a method which can perform "variable selection". Returning to the mice example from before, we see that the trace plot (Figure 4) shows that as λ decreases, more elements of $\hat{\beta}_\lambda^{\text{lasso}}$ become nonzero.

```
library(glmnet)
library(spls)
data(mice)
set.seed(1)
fit <- glmnet(x = mice$x[,1:10], y = mice$y[,1], alpha = 1)
plot(fit, xvar="lambda")
```

At first glance, it seems incredible that replacing the squared euclidean norm with the sum of absolute values would lead to "sparse" estimates (i.e., estimates which have many entries equal to zero). However, there is a very simple picture, taken from Hastie et al. (2009), which lends intuition for this phenomenon. Before we examine this figure, let us first point out that we can write the lasso equivalently in its *constrained* form.

$$\hat{\beta}_t^{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 \right\} \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t. \tag{2}$$

The formulation we described above with $\hat{\beta}_\lambda^{\text{lasso}}$ is sometimes called the *Lagrangian form* of (2). Letting $t \to \infty$, we go from $\hat{\beta}_t^{\text{lasso}} \approx 0$ to $\hat{\beta}_t^{\text{lasso}} = \hat{\beta}$, the least squares estimator.

Now, returning to the question at hand: why does this approach lead to sparse estimates? In the figure above, we see the red ellipses are the contours of the sum-of-squares. The blue regions correspond to the constraints on β, i.e., $|\beta_1| + |\beta_2| < t$ from (2). The lasso solution is the point at which the contours of the sum-of-squares hit the blue constraint region. At left, we see this occurs at a solution where $\beta_1 = 0$, but $\beta_2 > 0$. At right, we display the constraint region for the constrained version of ridge regression, and see that the counters hit the constraint region at a point where neither coefficients are zero.

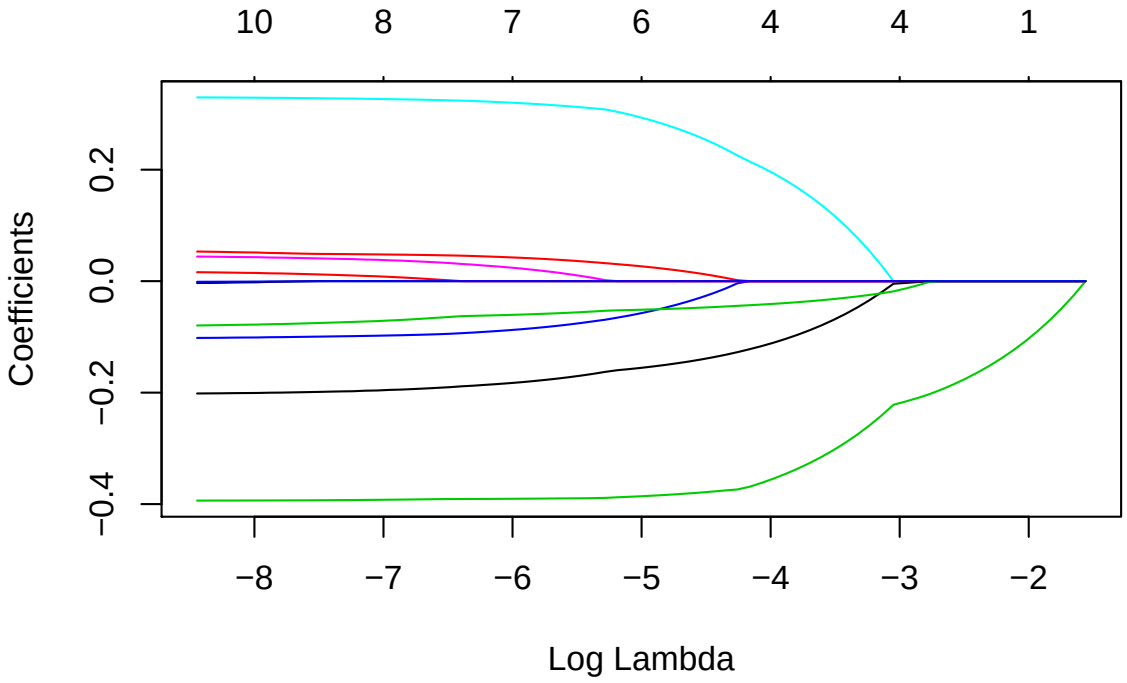

Figure 4: The output of plotting the `glmnet` object. Displayed is the "solution path" or "trace plot", i.e., the values of the estimated coefficients as a function of λ.

More generally, the constraint set $\sum_{j=1}^{p} |\beta_j| < t$ is a *cross-polytope*. In two dimensions, this is a square (as we see in Figure 5). In three dimensions, this is a octahedron. The key to achieving sparsity is that the vertices of the cross-polytope are unit vectors pointing along each coordinate axis, i.e., all possible permutations of the p elements $\{\pm 1, 0, 0, \ldots, 0\}$.

4.2 Digression on alternative methods for variable selection

There exist many methods for variable selection in linear regression. Perhaps the most well known are forward and backward stepwise regression; and best-subset regression. Forward stepwise regression sequentially adds predictors to the model using ordinary least squares. Backward elimination starts from a saturated model and removes predictors sequentially. These are both "greedy" approaches in the sense that they do not consider the entire space of models (of which there are 2^p).

Best subset selection, on the other hand, seeks to find the best model which has at most s nonzero coefficients:

$$\arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 \right\}, \quad \text{subject to} \quad \sum_{j=1}^{p} \mathbf{1}(\beta_j \neq 0) \leq s.$$

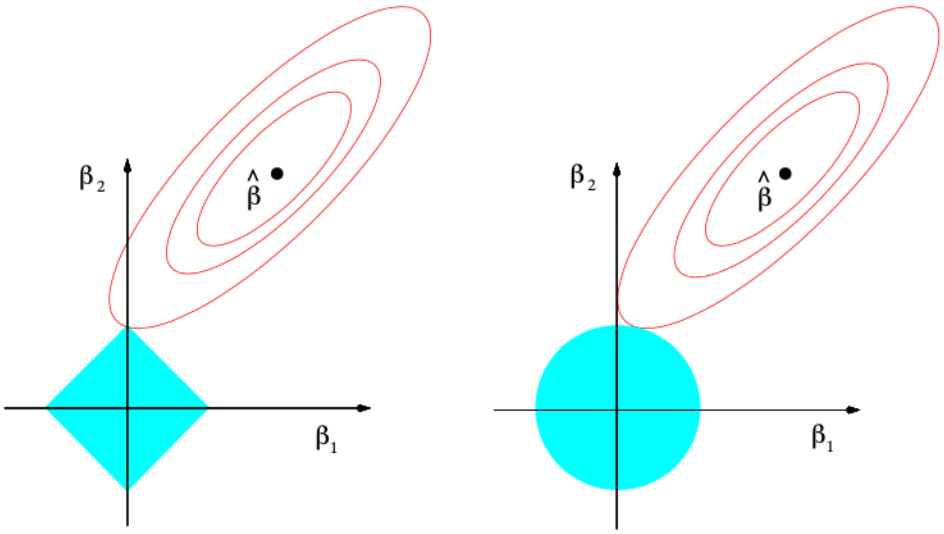

Figure 5: Counters of the residual sum of squares (in red) and the constraint sets (left) for lasso and (right) for ridge regression. Figure taken from Hastie et al. (2009).

The best-subset selection problem is NP-hard, so finding a global minimizer is computationally intensive. Until recently, the most efficient approach was the *leaps and bounds* algorithm, which could handle $p \approx 30 - 40$ in a reasonable amount of time. In 2016, however, Bertsimas et al. (2016) showed the best subset selection problem could be framed as a mixed integer optimization (MIO), for which there exist many efficient solvers. In minutes, they were able to compute global solutions for p in the 100s, and nearly global solutions for p in the 1000s. This is a promising development, and may be an interesting topic for a course project.

4.3 Computation

4.3.1 Lasso solution under orthonormal design

To get a sense of how the lasso operates, it is instructive to consider the lasso under "orthonormal design", which means that the design matrix satisfies $\mathbf{X}'\mathbf{X} = I_p$. In general, we see that the first order optimality conditions for the lasso are

$$\mathbf{X}'\mathbf{X}\hat{\beta}_\lambda^{\text{lasso}} - \mathbf{X}'\mathbf{y} + \lambda\partial\|\hat{\beta}_\lambda^{\text{lasso}}\|_1 \ni 0$$

where $\partial\|\beta\|_1$ denotes the *subdifferential* (i.e., set of all subgradients) of $\|\beta\|_1 := \sum_j |\beta_j|$ evaluated at β.

Subdifferential

The subdifferential ∂f of a convex function f at x is the set of all vectors g satisfying

$$f(y) \geq f(x) + g'(y - x) \quad \forall y.$$

A vector g satsifying this inequality is said to be a *subgradient* of f at x. Clearly, if f is differentiable at x, this set is the singleton $\nabla f(x)$.

We can check that

$$[\partial \|\beta\|_1]_j = \begin{cases} 1 & \beta_j > 0 \\ -1 & \beta_j < 0 \\ [-1, 1] & \beta_j = 0 \end{cases}, \quad (j = 1, \ldots, p).$$

Thus, under orthonormal design, we have *zero subgradient equation*

$$\hat{\beta}_\lambda^{\text{lasso}} - \mathbf{X}'\mathbf{y} + \lambda s = 0$$

for some $s \in \partial \|\hat{\beta}_\lambda^{\text{lasso}}\|_1$. Focusing on the jth component of $\hat{\beta}_\lambda^{\text{lasso}}$, we see that

$$[\hat{\beta}_\lambda^{\text{lasso}}]_j + \lambda s_j = \mathbf{X}_j'\mathbf{y}$$

We will now derive a closed form for $[\hat{\beta}_\lambda^{\text{lasso}}]_j$ under orthonormal design.

First, suppose that $\mathbf{X}_j'\mathbf{y} - \lambda > 0$. Then, if we set $[\hat{\beta}_\lambda^{\text{lasso}}]_j = \mathbf{X}_j'\mathbf{y} - \lambda$, $s_j = 1$ since $[\hat{\beta}_\lambda^{\text{lasso}}]_j > 0$. Putting this all together, we would have

$$[\hat{\beta}_\lambda^{\text{lasso}}]_j + \underbrace{\lambda s_j}_{=\lambda} = \mathbf{X}_j'\mathbf{y},$$

which are exactly the first order conditions. A similar argument can be used to show that if $\mathbf{X}_j'\mathbf{y} + \lambda < 0$, $[\hat{\beta}_\lambda^{\text{lasso}}]_j = \mathbf{X}_j'\mathbf{y} + \lambda$ also satisfies the first order conditions. Finally, if $|\mathbf{X}_j'\mathbf{y}| < \lambda$, we can set $[\hat{\beta}_\lambda^{\text{lasso}}]_j = 0$ and take $s_j = \mathbf{X}_j'\mathbf{y}/\lambda$, which is contained in $[-1, 1]$ by assumption. Thus, we have

$$\underbrace{[\hat{\beta}_\lambda^{\text{lasso}}]_j}_{=0} + \lambda(\mathbf{X}_j'\mathbf{y}/\lambda) = \mathbf{X}_j'\mathbf{y} \implies [\hat{\beta}_\lambda^{\text{lasso}}]_j + \lambda s_j = \mathbf{X}_j'\mathbf{y},$$

verifying that $[\hat{\beta}_\lambda^{\text{lasso}}]_j = 0$ satisfies the optimality conditions.

Putting this all together, we have

$$[\hat{\beta}_\lambda^{\text{lasso}}]_j = \begin{cases} \mathbf{X}_j'\mathbf{y} - \lambda & : \mathbf{X}_j'\mathbf{y} > \lambda \\ \mathbf{X}_j'\mathbf{y} + \lambda & : \mathbf{X}_j'\mathbf{y} < -\lambda \\ 0 & : |\mathbf{X}_j'\mathbf{y}| \leq \lambda \end{cases}, \quad j = 1, \ldots, p$$

$$\implies [\hat{\beta}_\lambda^{\text{lasso}}]_j = \text{soft}(\mathbf{X}_j'\mathbf{y}, \lambda) := \max(|\mathbf{X}_j'\mathbf{y}| - \lambda, 0)\text{sign}(\mathbf{X}_j'\mathbf{y}), \quad j = 1, \ldots, p$$

where soft is often referred to as the "soft-thresholding operator". We can see how this functions by examining the following plot.

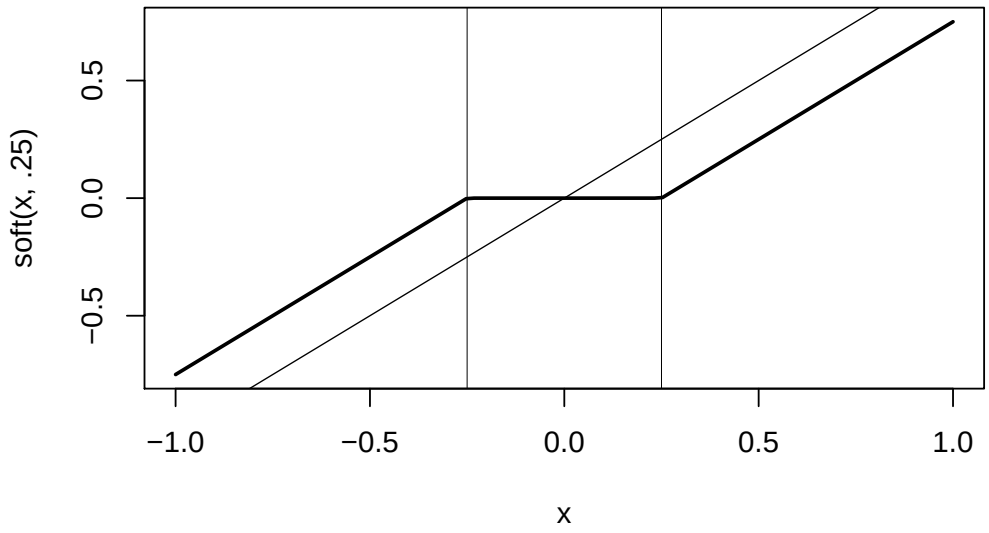

Figure 6: The soft-thresholding operator with threshold 0.25 (thick line) and the $y = x$ line (thin line).

4.3.2 Coordinate descent algorithm

The orthonormal design setting is instructive as an example, but not generally useful in practice. To solve the lasso optimization problem in more general settings, the most commonly used approach is componentwise "coordinate descent". We will outline the algorithm and derive its steps.

In brief, coordinate descent operates by

1. Fixing all but the jth optimization variable and minimizing the criterion with respect to the jth variable.
2. Repeating Step 1 for $j = 1, \ldots, p$ iteratively until the convergence (e.g., until the KKT conditions are approximately satisfied).

Specifically, let

$$f(\beta_j; \tilde{\beta}_{-j}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}_{-j}\tilde{\beta}_{-j} - \mathbf{X}_j\beta_j\|_2^2 + \lambda|\beta_j|.$$

The crux of coordinate descent is solving the problem $\arg\min_{\beta_j \in \mathbb{R}} f(\beta_j; \tilde{\beta}_{-j})$ Notice, if we let $r_j = \mathbf{y} - \mathbf{X}_{-j}\tilde{\beta}_{-j}$, we have

$$f(\beta_j; \tilde{\beta}_{-j}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}_{-j}\tilde{\beta}_{-j} - \mathbf{X}_j\beta_j\|_2^2 + \lambda|\beta_j| = \frac{1}{2}r_j'r_j - r_j'\mathbf{X}_j\beta_j + \beta_j'\mathbf{X}_j'\mathbf{X}_j\beta_j + \lambda|\beta_j|$$

Hence, β_j^+ minimizes $f(\beta_j; \tilde{\beta}_{-j})$ if

$$0 \in -r_j'\mathbf{X}_j + \mathbf{X}_j'\mathbf{X}_j'\beta_j^+ + \lambda\partial|\beta_j^+| \iff 0 \in -\frac{r_j'\mathbf{X}_j}{\mathbf{X}_j'\mathbf{X}_j} + \beta_j^+ + \frac{\lambda}{\mathbf{X}_j'\mathbf{X}_j}\partial|\beta_j^+|$$

so that by the same arguments used to derive the soft threholding operator under orthonormal design

$$\beta_j^+ = \text{soft}\left(\frac{r_j'\mathbf{X}_j}{\mathbf{X}_j'\mathbf{X}_j}, \frac{\lambda}{\mathbf{X}_j'\mathbf{X}_j}\right).$$

Thus, the cyclical coordinate descent algorithm can be summarized as follows.

Algorithm 1: Coordinate descent algorithm for the lasso

1. Initialize $\tilde{\beta} = (\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_p)$.
2. Compute $r = \mathbf{y} - \mathbf{X}\tilde{\beta}$
3. Repeat until convergence
   (a) For $j = 1, \ldots, p$ in sequence
      i. Compute $r_j = r + \mathbf{X}_j\tilde{\beta}_j$
      ii. Set $\tilde{\beta}_j = \text{soft}\left(\frac{r_j'\mathbf{X}_j}{\mathbf{X}_j'\mathbf{X}_j}, \frac{\lambda}{\mathbf{X}_j'\mathbf{X}_j}\right)$
      iii. Compute $r = r_j - \mathbf{X}_j\tilde{\beta}_j$

4.3.3 Implementation "tricks"

There are a number of useful tricks for computing the lasso, or its so-called "solution path" (i.e., $\beta_\lambda^{\text{lasso}}$ corresponding to many, ordered values of λ). Here, I will mention these and provide references for further reading.

- **Candidate tuning parameters**: Without any guidance, it is often difficult to determine a reasonable set of candidate tuning parameters. The most principled approach is as follows: (i) find the tuning parameter value λ_{max} which is the smallest that ensures $\hat{\beta}_{\lambda_{\text{max}}}^{\text{lasso}} = 0$; (ii) consider tuning parameters equally spaced on the log-10-scale between λ_{max} and $\delta\lambda_{\text{max}}$ for some $\delta \in (0, 1)$, e.g., $\delta = 0.1$.

  To find the tuning parameter yielding complete sparsity, we need only check for what λ would the first order conditions at $\hat{\beta}_\lambda^{\text{lasso}} = 0$ be satisfied. From before, we saw

$$\mathbf{X}'\mathbf{X}\hat{\beta}_\lambda^{\text{lasso}} - \mathbf{X}'\mathbf{y} + \lambda\partial\|\beta_\lambda^{\text{lasso}}\| \ni 0 \tag{3}$$

so that if $\hat{\beta}_\lambda^{\text{lasso}} = 0$, we have first order conditions

$$-\mathbf{X}'\mathbf{y} + \lambda\partial\|0\|_1 \ni 0$$

where $\partial\|0\|_1 = [-1,1] \times [-1,1] \times \cdots \times [-1,1]$. Hence, if we set $\lambda = \max_j |\mathbf{X}_j'\mathbf{y}|$, then by setting each $s_j = \mathbf{X}_j'\mathbf{y}/\lambda \in [-1,1]$, $s \in \partial\|0\|_1$ and thus, the zero subgradient equation

$$-\mathbf{X}'\mathbf{y} + \lambda s = 0$$

is satisfied and $\hat{\beta}_\lambda^{\text{lasso}} = 0$ is optimal. Thus, for the lasso,

$$\lambda_{\max} \geq \max_j |\mathbf{X}_j'\mathbf{y}|$$

is sufficient to ensure that $\hat{\beta}_\lambda^{\text{lasso}} = 0$.

- **Warm starting**: This is a method for computing the entire solution path over a sequence of candidate tuning parameters. If these candidate tuning parameters are $\lambda_1 > \lambda_2 > \cdots > \lambda_M$, warm starting referees to initializing the coordinate descent algorithm for computing the estimator with tuning parameter λ_j at the solution obtained with tuning parameter λ_{k-1}. Because, $\lambda_j \approx \lambda_{k-1}$, we know $\hat{\beta}_{\lambda_j}^{\text{lasso}} \approx \hat{\beta}_{\lambda_{k-1}}^{\text{lasso}}$, so if we initialize the algorithm for computing $\hat{\beta}_{\lambda_k}^{\text{lasso}}$ at $\hat{\beta}_{\lambda_{k-1}}^{\text{lasso}}$, we should require much fewer iterations than if we initialized at, say, the zeros vector. For this reason, many packages encourage users to provide many tuning parameter values rather than a single value. It is often the case that computing the entire solution path can be faster than computing just $\hat{\beta}_\lambda^{\text{lasso}}$ with λ relatively close to zero.

- **Strong rule** This approach improves on warm-starting by guessing (in a principled way) which predictors will still have coefficients equal to zero for tuning parameter λ_k given the coefficient estimates for λ_{k-1}.

  Strong-rule for discarding predictors (Tibshirani et al., 2012)

  Given $\hat{\beta}_{\lambda_{k-1}}^{\text{lasso}}$, the lasso estimator with tuning parameter λ_{k-1}, the strong rule discards the jth predictor from the optimization problem for computing $\hat{\beta}_{\lambda_k}^{\text{lasso}}$ if

  $$|\mathbf{X}_j'(\mathbf{y} - \mathbf{X}\hat{\beta}_{\lambda_{k-1}}^{\text{lasso}})| < 2\lambda_k - \lambda_{k-1}.$$

- **The active set trick**: The strong-rule is not foolproof: it can incorrectly remove predictors in special cases, and almost always retains predictors whose coefficient estimates will be zero. For this reason, the strong rule is used in combination with the so-called *active set trick*.

First, we run one pass of coordinate descent on all variables which may have nonzero coefficients as indicated by the strong rule. After this first pass, we only run coordinate descent on those coefficients whose iterates are currently nonzero. At convergence, we run one pass of coordinate descent on discarded predictors: if no new predictors enter, we terminate the algorithm; if a new predictor enters, add the new predictor to the "active set" and repeat.

To terminate the overall algorithm, we check whether (3) are satisfied. If not, we can determine which coordinate of $\tilde{\beta}_j$ violates the conditions: we then add the corresponding predictor to the current active set and repeat.

With these tricks, we can improve on the coordinate descent algorithm as follows.

Algorithm 2: Coordinate descent algorithm for the lasso with $\lambda = \lambda_k$

1. Initialize $\tilde{\beta} = \hat{\beta}^{\text{lasso}}_{\lambda_{k-1}}$. `# warm-start`
2. Compute $r = \mathbf{y} - \mathbf{X}\tilde{\beta}$
3. $\mathcal{A} = \{j : |\mathbf{X}_j' r| \geq 2\lambda_k - \lambda_{k-1}\}$ `# strong-rule`
4. $\tilde{\beta}_k = 0$ for all $k \in \{1, \ldots, p\} \setminus \mathcal{A}$
5. For $l \in \mathcal{A}$
   (a) Compute $r_l = r + \mathbf{X}_l \tilde{\beta}_l$
   (b) Compute $\beta^+ = \text{soft}\left(\frac{r_l' \mathbf{X}_l}{\mathbf{X}_l' \mathbf{X}_l}, \frac{\lambda}{\mathbf{X}_l' \mathbf{X}_l}\right)$
   (c) Set $[\tilde{\beta}]_l = \beta^+$
   (d) Compute $r = r_l - \mathbf{X}_l \tilde{\beta}_l$
6. $\tilde{A} = \{j : j \in \mathcal{A} \cap \tilde{\beta}_j \neq 0\}$
7. Repeat until convergence
   (a) For $l \in \tilde{\mathcal{A}}$ `# active-set trick`
      i. Compute $r_l = r + \mathbf{X}_l \tilde{\beta}_l$
      ii. Compute $\beta^+ = \text{soft}\left(\frac{r_l' \mathbf{X}_l}{\mathbf{X}_l' \mathbf{X}_l}, \frac{\lambda}{\mathbf{X}_l' \mathbf{X}_l}\right)$
      iii. Set $[\tilde{\beta}]_l = \beta^+$
      iv. Compute $r = r_l - \mathbf{X}_l \tilde{\beta}_l$
8. Rerun 5 once for all $l \in \mathcal{A}$. If any new coefficients become nonzero, add their indices to $\tilde{\mathcal{A}}$ and return to 7. If not, check whether $\tilde{\beta}$ satisfies the KKT conditions (3). If any component of $\tilde{\beta}$ violates the KKT conditions, add its index to $\mathcal{A}$ and $\tilde{\mathcal{A}}$ and return to 7.

4.4 LARS (path algorithm)

LARS (least angle regression) is an algorithm which is closely related to forward stepwise regression. Like forward stepwise regression: starts with no predictors, and first adds the predictor most correlated with $\mathbf{y}$, say $\mathbf{X}_j$. However, rather than fix the coefficient at the OLS estimate, LARS moves the coefficient continuously towards its OLS estimate (i.e., considering "shrunken" estimates). It proceeds as follows, taken closely from Hastie et al. (2009):

Algorithm 3: LARS algorithm

1. Standardize predictors to have mean zero and unit norm. Let $r = \mathbf{y}$, and set $\beta_1, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{X}_j$ most correlated with r.
3. Move β_j from 0 to its least squares estimate $\mathbf{X}_j' r$ until another predictor, say, $\mathbf{X}_k$ has as much correlation with the current residual as does $\mathbf{X}_j$.
4. Move β_j and β_k in the direction of the joint least squares estimator of the residual on $\mathbf{X}_j, \mathbf{X}_k$ until another competitor $\mathbf{X}_l$ has an much correlation with the current residual.
5. Repeat this procedure until $\min(n-1, p)$ predictors have entered the model.

Remarkably this algorithm almost exactly recovers the solution path of the lasso going from $\lambda_{\max} \to 0$. In the figure below, we show the solution paths for both LARS using Algorithm 3, and the lasso solution path. The only difference between the two occurs when a coefficient's value crosses zero.

With a slight modification, LARS can indeed recover the solution path for the lasso exactly.

Algorithm 4: LARS algorithm for lasso solution path

1. Standardize predictors to have mean zero and unit norm. Let $r = \mathbf{y}$, and set $\beta_1, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{X}_j$ most correlated with r.
3. Move β_j from 0 to its least squares estimate $\mathbf{X}_j' r$ until another predictor, say, $\mathbf{X}_k$ has as much correlation with the current residual as does $\mathbf{X}_j$.
4. Move β_j and β_k in the direction of the joint least squares estimator of the residual on $\mathbf{X}_j, \mathbf{X}_k$ until another competitor $\mathbf{X}_l$ has an much correlation with the current residual.

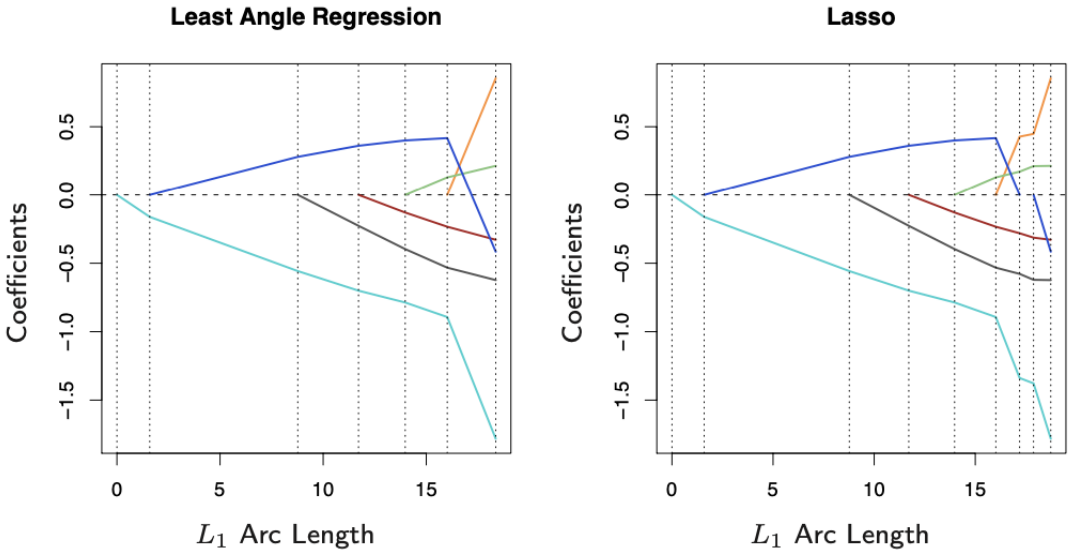

Figure 7: The LARS solution path (using Algorithm 3) and the lasso solution path as a function of the L_1 arc-length. Figure taken from Hastie et al. (2009).

(a) **If a non-zero coefficient hits zero, remove this predictor and recompute the joint least squares direction with predictors remaining.**

5. Repeat this procedure until $\min(n-1, p)$ predictors have entered the model.

Let us now briefly discuss the details of this procedure. Suppose that at the current step, we have a set of variables denoted by $\mathcal{A} \subset \{1, \dots, p\}$ in our current model; and let $\beta_{\mathcal{A}}$ be the coefficient estimate at this step. If we let $r = \mathbf{y} - \mathbf{X}_{\mathcal{A}}\beta_{\mathcal{A}}$ be the current residual, we move in the direction

$$\delta = (\mathbf{X}'_{\mathcal{A}}\mathbf{X}_{\mathcal{A}})^{-1}\mathbf{X}'_{\mathcal{A}}r,$$

i.e., our next coefficient estimate will have the form $\beta^+_{\mathcal{A}} = \alpha\delta + \beta_{\mathcal{A}}$ for some $\alpha \in (0, 1)$ chosen so that some variable not belonging to $\mathcal{A}$ is equally correlated with $r = y - X_{\mathcal{A}}(\alpha\delta + \beta_{\mathcal{A}})$ as are the variables in $\mathcal{A}$: Efron et al. (2004) derives exact formulas for such an α.

While we will not prove this, the idea is that by choosing the direction δ in which to move, the correlations between all the variables in $\mathcal{A}$ and r are exactly equivalent and decreasing as $\alpha \to 1$.

4.5 Variants of the lasso

In the next section, we will show that the lasso has nice statistical properties. Before that, however, we should adress some "deficiencies" of the lasso.

1. When two predictors are highly correlated, lasso tends to select one and exclude the other (essentially arbitrarily). Moreover, by construction, lasso can have at most n nonzero coefficient estimates.

2. Even as $n \to \infty$ with p fixed, the lasso coefficient estimates remained biased for "large" coefficient values. For example, in the orthonormal design setting

$$\mathrm{E}|[\hat{\beta}_\lambda^{\text{lasso}}]_j - \beta_j^*]| = \lambda \text{ if } |\beta_j^*| > \lambda.$$

4.5.1 Elastic net

Fortunately, there exist variations of the lasso which address both of these issues. A method addressing the first is the *elastic net*, which is defined as

$$\arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^p |\beta_j| + \frac{\gamma}{2} \sum_{j=1}^p \beta_j^2 \right\}.$$

As we can see, the elastic net balances a ridge penalty and lasso penalty. This method addresses both issues mentioned in 1) above. When two variables are highly correlated, elastic net shrinks both coefficeints together: lasso just selects one (e.g., consider how two correlated variables would enter in the LARS algorithm). In addition, the inclusion of a ridge penalty allows for more than $n - 1$ nonzero coefficient estimates. Fortunately, the elastic net is just as easy to compute as lasso via coordinate descent. The coordinate-wise update (e.g., from Algorithm 1) is nearly as simple as for lasso.

Exercise. Derive the coordinate-wise update for the elastic net version of 3(a)ii of Algorithm 1.

4.5.2 Adaptive lasso

There are a number of methods which address point 2) taking slightly different approaches. The adaptive lasso is one such approach. This estimator penalizes each coefficient by its own tuning parameter

$$\arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^p w_j|\beta_j| \right\},$$

where $w_j \geq 0$ are user specified weights for $j = 1, \ldots, p$. Ideally, we would want $w_j = 0$ for j where $\beta_j^* \neq 0$ and $w_j = \infty$ otherwise (were this information known apriori). In practice, we try for w_j small for large β_j^* and vice versa. One way to achieve this is to use $w_j = |\tilde{\beta}_j|^{-1}$ where $\tilde{\beta}$ is some initial estimate of β (e.g., using ridge regression), however any nonincreasing function $w(\tilde{\beta})$ could yield a reasonable set of weights, given the context.

4.5.3 Folded concave penalties

To deal with the bias of large coefficient estimates, an alternative is to use a penalty which "flattens" as β_j, the jth optimization variable, increases. Many folded concave penalties exist, but the two most popular are SCAD (smoothly clipped absolute deviations) and MCP (minimax concave penalty).

SCAD replaces $\lambda|\beta_j|$ in the lasso criterion with

$$P_1(\beta_j; \lambda, \gamma) = \begin{cases} \lambda|\beta_j| & : |\beta_j| \leq \lambda \\ \frac{2\gamma\lambda|\beta_j| - \beta_j^2 - \lambda^2}{2(\gamma-1)} & : \lambda < |\beta_j| < \gamma\lambda \\ \frac{\lambda^2(\gamma+1)}{2} & : |\beta_j| \geq \gamma\lambda \end{cases}$$

for tuning parameter $\gamma > 2$. Similarly, MCP replaces $\lambda|\beta_j|$ in the lasso criterion with

$$P_2(\beta_j; \lambda, \gamma) = \begin{cases} \lambda|\beta_j| - \frac{\beta_j^2}{2\gamma} & : |\beta_j| \leq \gamma\lambda \\ \frac{\gamma\lambda^2}{2} & : |\beta_j| > \gamma\lambda \end{cases}$$

for tuning parameter $\gamma > 1$.

In the Figure 8, we examine the penalties and corresponding thresholding operators (i.e., the solution in the orthonormal case with $p = 1$) for lasso (soft-thresholding described before), SCAD, and MCP. In the left plot, We see that as β_j gets larger, the value of the penalty under SCAD and MCP becomes constant. In the right plot, we see that as the input gets larger, both thresholding operators for SCAD and MCP become equal to the identity, whereas lasso remains shrunken towards zero. Of course, there is a price to pay for using SCAD and MCP: both penalties are nonconvex, and there is a second parameter, γ to be tuned.

5 Statistical properties of the lasso

5.1 Consistency

In this section, we will study the statistical properties of the lasso. Traditional asymptotics focus on the properties of estimators as $n \to \infty$ with p fixed. Since lasso is often used in high-dimensional settings, the most useful statistical theory allows p to grow with n.

For the remainder of this section, we will make the following assumptions about the data generating model:

1. The design matrix $X = (x_1, \ldots, x_n)' \in \mathbb{R}^{n \times p}$ is non-random and has columnwise squared euclidean norm n (i.e., $\|X_j\|_2^2 = n$) and average zero.

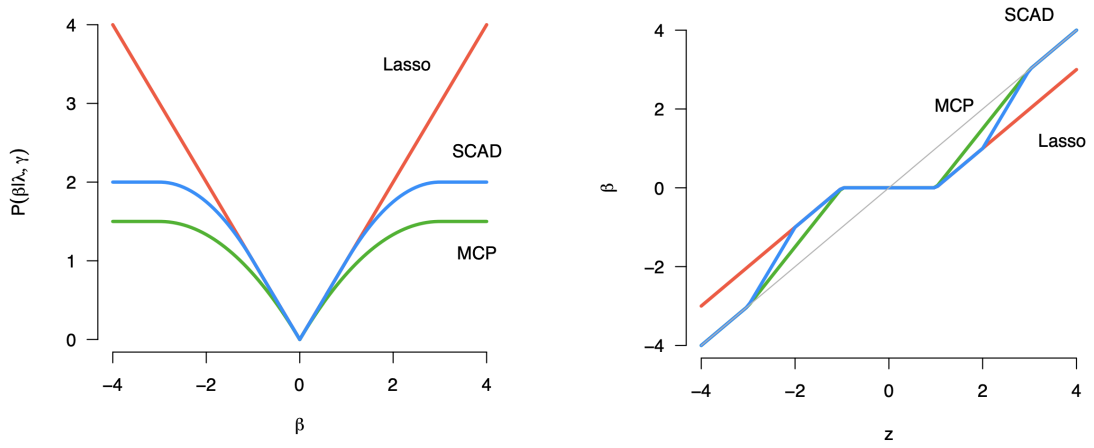

Figure 8: (Left) The penalties applied elementwise under lasso, SCAD, and MCP. (Right) The thresholding operators used under lasso, SCAD, and MCP. Both figures are from Patrick Breheny's lecture notes.

2. The data are generated from the model $Y_i = x_i'\beta^* + \epsilon_i$ where ϵ_i has mean zero, is σ-subgaussian (e.g., a mean zero normal random variable with variance σ^2 is σ-subgaussian), and ϵ_i and ϵ_j are independent and identically distributed for $i \neq j$. Hence, we can write
$$y = X\beta^* + \epsilon. \tag{4}$$

3. The coefficient matrix β^* has s nonzero entries where $s \ll p$. We will let $\mathcal{S}$ denote the support of β^*:
$$\mathcal{S} = \{j : \beta_j^* \neq 0\}, \quad \mathcal{S}^c = \{1, \ldots, p\} \setminus \mathcal{S}.$$

To make matters as concrete as possible, we redefine the lasso so that we divide the sum of squares by $2n$:
$$\hat{\beta}_\lambda^{\text{lasso}} = \arg\min_{\beta \in \mathbb{R}^p} \left\{ \underbrace{\frac{1}{2n}\|y - X\beta\|_2^2}_{g(\beta)} + \lambda\|\beta\|_1 \right\}$$
where $\|A\|_1 = \sum_j |a_j|$. Throughout this section, we will use the notation $\beta_{\mathcal{S}}^*$ and $[\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}}$ to denote the subvectors of β^* and $\hat{\beta}_\lambda^{\text{lasso}}$ consisting of all elements whose indices belong to $\mathcal{S}$.

The main ingredient of our proof will be the establishment of the so-called "basic inequality". First, because $\hat{\beta}_\lambda^{\text{lasso}}$ is the global minimizer of the lasso criterion defined above, we know
$$g(\hat{\beta}_\lambda^{\text{lasso}}) + \lambda\|\hat{\beta}_\lambda^{\text{lasso}}\|_1 \leq g(\beta^*) + \lambda\|\beta^*\|_1$$

by definition. Hence,

$$\begin{aligned} g(\hat{\beta}_\lambda^{\text{lasso}}) - g(\beta^*) &\leq \lambda\|\beta^*\|_1 - \lambda\|\hat{\beta}_\lambda^{\text{lasso}}\|_1 \\ &= \lambda\|\beta^*_{\mathcal{S}}\|_1 - \lambda\|\hat{\beta}_\lambda^{\text{lasso}}\|_1 \\ &= \lambda\|\beta^*_{\mathcal{S}} - [\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}} + [\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}}\|_1 - \lambda\|\hat{\beta}_\lambda^{\text{lasso}}\|_1 \\ &= \lambda\|\beta^*_{\mathcal{S}} - [\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}} + [\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}}\|_1 - \lambda\|[\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}}\|_1 - \lambda\|[\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}^c}\|_1 \\ &\leq \lambda\|\beta^*_{\mathcal{S}} - [\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}}\|_1 - \lambda\|[\hat{\beta}_\lambda^{\text{lasso}}]_{\mathcal{S}^c}\|_1 \end{aligned}$$

where the last line follows from the triangle inequality. Thus, letting $\hat{\Delta} = \beta^* - \hat{\beta}_\lambda^{\text{lasso}}$,

$$g(\hat{\beta}_\lambda^{\text{lasso}}) - g(\beta^*) \leq \lambda\left(\|\hat{\Delta}_{\mathcal{S}}\|_1 - \|\hat{\Delta}_{\mathcal{S}^c}\|_1\right).$$

Moreover,

$$\begin{aligned} g(\hat{\beta}_\lambda^{\text{lasso}}) - g(\beta^*) &= \frac{1}{2n}\|y - X\hat{\beta}_\lambda^{\text{lasso}}\|_2^2 - \frac{1}{2n}\|y - X\beta^*\|_2^2 \\ &= \frac{1}{2n}\|X\beta^* + \epsilon - X\hat{\beta}_\lambda^{\text{lasso}}\|_2^2 - \frac{1}{2n}\|\epsilon\|_2^2 \\ &= \frac{1}{2n}\|X\hat{\Delta} + \epsilon\|_2^2 - \frac{1}{2n}\|\epsilon\|_2^2 \\ &= \frac{1}{2n}\hat{\Delta}'X'X\hat{\Delta} + \frac{1}{n}\hat{\Delta}'X'\epsilon \\ &\geq \frac{1}{2n}\|X\hat{\Delta}\|_2^2 - \frac{1}{n}|\hat{\Delta}'X'\epsilon| \\ &\geq \frac{1}{2n}\|X\hat{\Delta}\|_2^2 - \frac{1}{n}\|\hat{\Delta}\|_1\|X'\epsilon\|_\infty \end{aligned}$$

where $\|A\|_\infty = \max_j |A_j|$ and the final line follows from an application of Holders inequality. To conclude, we have a variation of the so-called "basic inequality".

The basic inequality for lasso

For every $\lambda > 0$, n, p, and support set $\mathcal{S}$,

$$\frac{1}{2n}\|X\hat{\Delta}\|_2^2 - \frac{1}{n}\|\hat{\Delta}\|_1\|X'\epsilon\|_\infty \leq \lambda\left(\|\hat{\Delta}_{\mathcal{S}}\|_1 - \|\hat{\Delta}_{\mathcal{S}^c}\|_1\right)$$

where $\hat{\Delta} = \beta^* - \hat{\beta}_\lambda^{\text{lasso}}$.

To use the basic inequality to establish high probability bounds for $\|\Delta\|_2$, we will introduce an assumption, A1, and an event E1:

- A1: There exists constant $\kappa > 0$ such that $\frac{1}{2n}\|X\Delta\|_2^2 \geq \kappa\|\Delta\|_2^2$ for all $\Delta \in \mathbb{R}^p$.
- E1: $\lambda \geq \frac{2}{n}\|X'\epsilon\|_\infty$

In a subsequent section, we will replace A1 with something that can be more reasonably assumed under E1, and will assign a probability to E1 for a particular choice of λ.

Because we have $\frac{1}{2n}\|X\hat{\Delta}\|_2^2 - \frac{1}{n}\|\hat{\Delta}\|_2\|X'\epsilon\|_\infty - \lambda\left(\|\hat{\Delta}_{\mathcal{S}}\|_1 - \|\hat{\Delta}_{\mathcal{S}^c}\|_1\right) \leq 0$, on A1 $\cap$ E1, we also have

$$\begin{aligned}
&\kappa\|\hat{\Delta}\|_2^2 - \frac{\lambda}{2}\|\hat{\Delta}\|_1 - \lambda\left(\|\hat{\Delta}_{\mathcal{S}}\|_1 - \|\hat{\Delta}_{\mathcal{S}^c}\|_1\right) \leq 0 \\
\implies &\kappa\|\hat{\Delta}\|_2^2 - \frac{\lambda}{2}(\|\hat{\Delta}_{\mathcal{S}}\|_1 + \|\hat{\Delta}_{\mathcal{S}^c}\|_1) - \lambda\left(\|\hat{\Delta}_{\mathcal{S}}\|_1 - \|\hat{\Delta}_{\mathcal{S}^c}\|_1\right) \leq 0 \\
\implies &\kappa\|\hat{\Delta}\|_2^2 - \frac{3\lambda}{2}(\|\hat{\Delta}_{\mathcal{S}}\|_1) \leq 0 \\
\implies &\kappa\|\hat{\Delta}\|_2^2 \leq \frac{3\lambda\sqrt{s}}{2}\|\hat{\Delta}\|_2
\end{aligned}$$

where the fourth line follows from the fact that $\|\hat{\Delta}_{\mathcal{S}}\|_1 \leq \sqrt{s}\|\hat{\Delta}\|_2$ when $\mathcal{S}$ has cardinality s. Finally, recalling that $\hat{\Delta} = \beta^* - \hat{\beta}_\lambda^{\text{lasso}}$, we have

$$\|\hat{\beta}_\lambda^{\text{lasso}} - \beta^*\|_2 \leq \frac{3\lambda\sqrt{s}}{2\kappa}. \tag{5}$$

Great! Now, we simply need to break down event E1 and assumption A1. Let us start with E1. We want to be able to control the probability of the event $\lambda \geq 2n^{-1}\|X'\epsilon\|_\infty$ by selecting λ as small as possible such that E1 still holds with high probability. To simplify matters, we assume that the ϵ_i are subgaussian, meaning that their tails behave like normal random variables. First, notice that

$$P\left(\frac{2}{n}\|X'\epsilon\|_\infty > \lambda\right) = P\left(\max_j |X_j'\epsilon| > \frac{n\lambda}{2}\right) \leq \sum_{j=1}^{p} P\left(|X_j'\epsilon| > \frac{n\lambda}{2}\right) \tag{6}$$

by the union bound, i.e., $P(\cup_i C_i) \leq \sum_i P(C_i)$ for events $C_1, C_2, \ldots$. Then, we will apply the following result.

Concentration of subgaussian random variables (Chernoff bound)

Let $u_1, \ldots, u_n$ be independent σ-subgaussian random variables. Then, for any $A \in \mathbb{R}^n$,

$$P\left(\left|\sum_{i=1}^{n} A_i u_i\right| > t\right) \leq 2\exp\left(-\frac{t^2}{2\sigma^2\|A\|_2^2}\right).$$

A direct application of the result yields:

$$P\left(|X_j'\epsilon| > \frac{n\lambda}{2}\right) \leq 2\exp\left(-\frac{n^2\lambda^2}{8\sigma^2\|X_j\|_2^2}\right) = 2\exp\left(-\frac{n\lambda^2}{8\sigma^2}\right)$$

since we normalized X_j so that $\|X_j\|_2^2 = n$. Hence, from (6), we have

$$P\left(\frac{2}{n}\|X'\epsilon\|_\infty \leq \lambda\right) \geq 1 - 2p \cdot \exp\left(-\frac{n\lambda^2}{8\sigma^2}\right)$$

which means that if we set $\lambda = k_1\sigma\{n^{-1}\log(p)\}^{1/2}$ for large constant k_1, it would follow that

$$\begin{aligned} P\left(\frac{2}{n}\|X'\epsilon\|_\infty \leq \lambda\right) &= P\left(\frac{2}{n}\|X'\epsilon\|_\infty \leq k_1\sigma\sqrt{\frac{\log p}{n}}\right) \\ &\geq 1 - 2p \cdot \exp\left\{-\frac{n}{8\sigma^2}\left(\frac{k_1^2\sigma^2\log(p)}{n}\right)\right\} = 1 - 2p^{1-\frac{k_1^2}{8}} \end{aligned}$$

which can be made arbitrarily close to one by taking k_1 sufficiently large. Thus, we know that if $\lambda = k_1\sigma\{n^{-1}\log(p)\}^{1/2}$, E1 will occur with probability at least $1 - 2p^{1-\frac{k_1^2}{8}}$. We now turn our attention to A1.

As stated, A1 is untenable in most applications wherein one would want to use the lasso. If $n < p$, no such κ can exist since there would trivially exist $\Delta \in \text{Null}(X)$. Thus, we need a more careful way to bound $\|X\hat{\Delta}\|_2^2$ in order to replace A1. To do this, we will instead assume that X satisfies the so-called "restricted eigenvalue condition".

Restricted eigenvalue condition

There exists a constant $\kappa(\mathcal{S}, c) > 0$ such that

$$\frac{1}{2n}\|X\Delta\|_2^2 \geq \kappa(\mathcal{S}, c)\|\Delta\|_2^2, \text{ for all } \Delta \in \{\Delta \in \mathbb{R}^p, \Delta \neq 0, \|\Delta_{\mathcal{S}^c}\|_1 \leq c\|\Delta_{\mathcal{S}}\|_1\}$$

for constant $c > 1$.

Notice that the restricted eigenvalue condition no longer requires the bound to hold for *any* Δ, but rather, only Δ nonzero belonging to the cone $\|\Delta_{\mathcal{S}^c}\|_1 \leq c\|\Delta_{\mathcal{S}}\|_1$. You should convince yourself that if the cardinality of $\mathcal{S}$, s, is small, and c is sufficiently large, this condition may not be so restrictive.

In order to apply the restricted eigenvalue condition, we have the following (deterministic) result, which we can prove using the basic inequality.

Result: If $\lambda \geq \frac{2}{n}\max_j |X_j'\epsilon|$, then $\hat{\Delta} \in \{\Delta \in \mathbb{R}^p : \Delta \neq 0, \|\Delta_{\mathcal{S}^c}\|_1 \leq 3\|\Delta_{\mathcal{S}}\|_1\}$.

Exercise. Prove the previous result using the basic inequality.

Hence, if event E1 occurs and we assume the restricted eigenvalue condition with constant $\kappa(\mathcal{S}, 3)$, it follows that on E1, $(2n)^{-1}\|\hat{\Delta}\|_2^2 \geq \kappa(\mathcal{S}, 3)\|\hat{\Delta}\|_2^2$, so we can replace A1 with the restricted eigenvalue condition and achieve (5). Putting all of the pieces together, we have the following theorem.

Lasso error bound

Assume X is non-random with columnwise euclidean norm $\sqrt{n}$, and that X satisfies the restricted eigenvalue condition with constant $\kappa(\mathcal{S}, 3)$. Assume also that $y = X\beta^* + \epsilon$ where the ϵ_i are independent, mean zero, σ-subgaussian random variables. If we set $\lambda = k_1\sigma\{n^{-1}\log(p)\}^{1/2}$, then

$$\|\hat{\beta}_\lambda^{\text{lasso}} - \beta^*\|_2 \leq \frac{3k_1\sigma}{2\kappa(\mathcal{S}, 3)}\sqrt{\frac{s\log p}{n}}$$

with probability at least $1 - 2p^{1-\frac{k_1^2}{8}}$.

This result is remarkable: it reveals that under some mild conditions,

$$\|\hat{\beta}_\lambda^{\text{lasso}} - \beta^*\|_2 = O_P\left(\sqrt{\frac{s\log p}{n}}\right).$$

Consider this: if we knew the important predictors ahead of time and used the least squares estimator, we would obtain an error bound of order $O_P(\sqrt{s/n})$. This means we nearly achieve this "oracle" error bound, paying only the penalty of the $\sqrt{\log p}$ factor for having to select from p predictors. Finally, this yields the following fact.

Consistency of lasso

Under the assumption that $\kappa(\mathcal{S}, 3)$ is bounded below for all (n, p), if $s\log p = o(n)$ and $\lambda \asymp n^{-1}\log p$, the lasso is consistent in the sense that $\|\hat{\beta}_\lambda^{\text{lasso}} - \beta^*\|_2 \to 0$ as $n \to \infty$.

6 Inference for the lasso

6.1 De-biased lasso

There are a number of methods for inference that can be used in combination with the lasso and its variants. We will focus on one approach, called *de-biasing*. The basic idea is quite simple: we will fit the lasso, then add back the bias induced by the lasso in order to obtain an estimator which has a known distribution (approximately). This section will follow closely from Hastie et al. (2015) and Javanmard and Montanari (2014).

First, recall that the OLS estimator $\hat{\beta}^{\text{OLS}} = (X'X)^{-1}X'y$ is unbiased for β^*. Thus, we can a "de-bias" the lasso estimator in the following way: let

$$\hat{\beta}^d = \hat{\beta}_\lambda^{\text{lasso}} + \frac{1}{n}\Theta X'(y - X\hat{\beta}_\lambda^{\text{lasso}})$$

be the "de-biased lasso" estimator where Θ is an approximate inverse of $\frac{1}{n}X'X$. Of course, if $n > p$ and $\frac{1}{n}X'X$ is non-singular, then $\hat{\beta}^d = \hat{\beta}^{\text{OLS}}$: in this sense, $\hat{\beta}^d$ is said to be de-biased. From this definition, since $y = X\beta^* + \epsilon$,

$$\begin{aligned}
\hat{\beta}^d &= \beta^* - \beta^* + \hat{\beta}_\lambda^{\text{lasso}} + \frac{1}{n}\Theta X'(y - X\hat{\beta}_\lambda^{\text{lasso}}) \\
&= \beta^* + I_p(\hat{\beta}_\lambda^{\text{lasso}} - \beta^*) + \frac{1}{n}\Theta X'(X\beta^* + \epsilon - X\hat{\beta}_\lambda^{\text{lasso}}) \\
&= \beta^* + \frac{1}{n}\Theta X'\epsilon + \underbrace{\left(\frac{1}{n}\Theta X'X - I_p\right)(\beta^* - \hat{\beta}_\lambda^{\text{lasso}})}_{\hat{\Gamma}}
\end{aligned}$$

If we assume that $\epsilon \sim \text{N}_n(0, \sigma^2 I_n)$, we have that

$$\hat{\beta}^d \xrightarrow{d} \text{N}_p\left(\beta^*, \frac{\sigma^2}{n^2}\Theta X'X\Theta\right) \quad \text{as } n \to \infty \tag{7}$$

as long as Θ is estimated in a manner such that $\|\hat{\Gamma}\|_\infty \to 0$ as $n \to \infty$. One approach is to estimate Θ column-by-column. Let $\hat{\Sigma} = \frac{1}{n}X'X$ and let m_j be defined as

$$m_j = \underset{m \in \mathbb{R}^p}{\arg\min}\, m'\hat{\Sigma}m, \quad \text{subject to} \quad \|\hat{\Sigma}m - e_j\|_\infty \leq \gamma$$

for small constant $\gamma > 0$ where e_j is the jth unit vector (i.e., is a vector of zeros in all but the jth position, which has a one). Then, with

$$\hat{\Theta} = (m_1, m_2, \ldots, m_p),$$

we have

$$\hat{\Sigma}\hat{\Theta} \approx I_p$$

and the variances for each $\hat{\beta}_j^d$ from (7) are relatively small. Based on the approximate distribution from (7), one can construct confidence intervals and perform standard hypothesis testing, as shown in the plot below, taken from Hastie et al. (2015).

This approach comes with some theoretical guarantees.

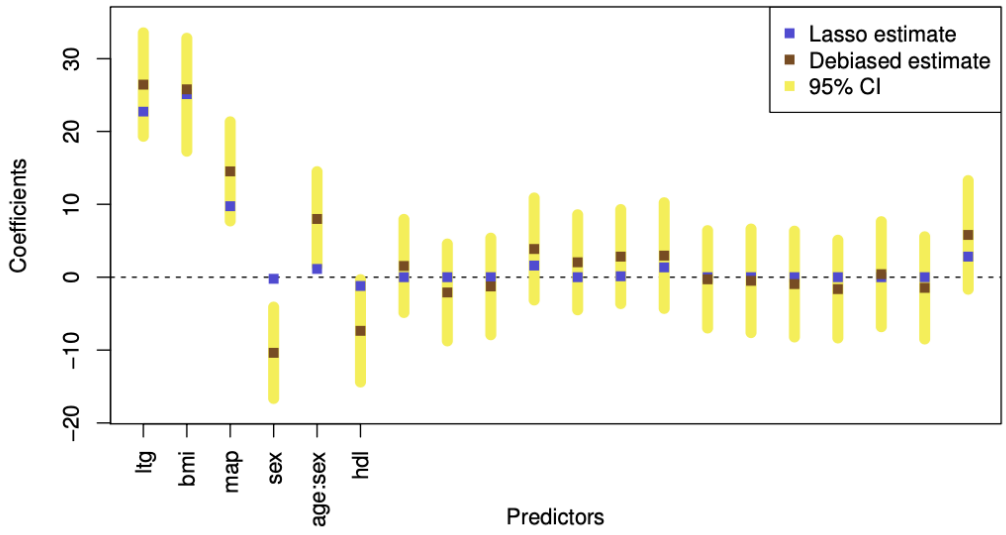

Figure 9: De-baised lasso confidence intervals, taken from Hastie et al. (2015).

Debiased lasso

Let X be a non-random matrix satisfying the *compatibility condition* (like restricted eigenvalue condition). Define the coherence parameter

$$\gamma_* = \min_{M \in \mathbb{R}^{p \times p}} \|M\Sigma - I\|_\infty.$$

Then, we have

$$\sqrt{n}(\hat{\beta}^d - \beta^*) = Z + \Lambda$$

where Z is gaussian and Λ is the bias satisfying

$$\|\Lambda\|_\infty \leq K\gamma_* \sigma s \sqrt{\log p}$$

with high probability.

6.2 Knockoffs

One recently proposed appraoch, the *knockoff filter*, can be used to control the false discovery rate, i.e., the expected proportion of all $\hat{\beta}_j$ estimated to be nonzero where in truth, $\beta_j^* = 0$. Formally, define

$$\text{FDR} = \text{E}\left[\frac{\#\left\{j : \beta_j^* = 0 \cap \hat{\beta}_j \neq 0\right\}}{\max\left(\#\left\{j : \hat{\beta}_j \neq 0\right\}, 1\right)}\right].$$

Throughout this section, we refer to the linear model from (4). To simplify notation, assume we have normalized the X_j so that $\|X_j\|_2^2 = X_j'X_j = 1$.

While we will focus on the application of knockoffs to the lasso, this framework is much more general than described here and can be applied an enormous number of models and methods. This section follows closely from Barber and Candès (2015).

6.2.1 Outline

The knockoff filter proceeds as follows:

1. **Construct knockoffs.** For every predictor X_j, construct a "knockoff" of X_j, $\tilde{X}_j$. We will do this for each X_j, so that we have $X \in \mathbb{R}^{n\times p}$ and $\tilde{X} \in \mathbb{R}^{n\times p}$. We want to construct the knockoffs so that

$$(\text{i})\tilde{X}'\tilde{X} = X'X, \quad (\text{ii})X'\tilde{X} = X'X - \text{Diag}(a)$$

   where $a \in \mathbb{R}^p$ is a non-negative vector. Due to (i), $\tilde{X}$ has the same covariance as X, and due to (ii), the correlation between X_j and $\tilde{X}_k$ is the same as the correlation between X_j and X_k for all $k \neq j$. However, we can see that

$$X_j'\tilde{X}_j = X_j'X_j - a_j = 1 - a_j, \quad j = 1, \ldots, p$$

   whereas $X_j'X_j = \tilde{X}_j'\tilde{X}_j = 1$. It is helpful to choose a_j as large as possible, which would mean that X_j is not too similar to $\tilde{X}_j$ (i.e., their correlation is small). We will return to the question of how to construct knockoffs so satsify (i) and (ii) later on. For now, suppose we have knockoffs $\tilde{X}$. Intuitively, because the knockoffs $\tilde{X}_j$ are constructed only using the original predictors X, they are only related to the response y through their correlation with the the original predictors.

2. **Calculate statistics for each pair of original and knockoff variables.** Next, we will compute statistics W_j for $j = 1, \ldots, p$ – each corresponding to a particular β_j^*. To do so, we will solve the following optimization problem

$$\hat{\beta} = \underset{\beta\in\mathbb{R}^{2p}}{\arg\min} \left\{ \frac{1}{2}\|y - \ddot{X}\beta\|_2^2 + \lambda \sum_{j=1}^{2p} |\beta| \right\}$$

   where $\ddot{X} = (X, \tilde{X}) \in \mathbb{R}^{n\times 2p}$. For each j, we have two statistics, Z_j and $\tilde{Z}_j$, defined as

$$Z_j = \sup\{\lambda : \hat{\beta}_j(\lambda) \neq 0\}, \quad \tilde{Z}_j = \sup\{\lambda : \hat{\beta}_{j+p}(\lambda) \neq 0\},$$

   for $j \in 1, \ldots, p$. Then, we construct our test statistic

$$W_j = \max(Z_j, \tilde{Z}_j)\text{sign}(Z_j - \tilde{Z}_j). \tag{8}$$

Under this construction, a large positive value of W_j indicates that X_j entered the model for a large value of λ, and before its knockoff $\tilde{X}_j$. Thus large W_j would suggest that X_j is truly related to y and thus belongs in the model. In general, the larger the W_j, the more evidence against the null hypothesis that $\beta_j^* = 0$.

3. **Calculate a data-dependent threshold for the statistics.** To control the FDR, we need to determine a cutoff wherein all W_j below this cutoff we will regard as potential false discoveries. That is, we want to determine a $t \geq 0$ such that if $W_j \geq t$, we would reject the hypothesis that $\beta_j^* = 0$. We define q as the target FDR and define a data-dependent threshold T as

$$T = \min \left\{ t \in \mathcal{W} : \frac{\#\{j : W_j \leq -t\}}{\#\max\left(\#\{j : W_j \geq t\}, 1\right)} \leq q \right\} \quad (T = +\infty \text{ if the set is empty}) \tag{9}$$

where $\mathcal{W} = \{|W_j| : j = 1, \ldots, p\} \setminus \{0\}$ is the set of unique nonzero values obtained by the $|W_j|$.

If we follow this procedure, we have the following result.

Modified FDR control with knockoffs

Let $\tilde{X}$ be a set of knockoffs satisfying (i) and (ii) and let W_j be as defined in (8). Define the estimated support as

$$\hat{\mathcal{S}} = \{j : W_j \geq T\}$$

where T is defined in (9). Then, for any $q \in [0, 1]$, the knockoff filter procedure satisfies

$$\mathrm{E}\left[\frac{\#\{j : \beta_j^* \neq 0 \cap j \in \hat{\mathcal{S}}\}}{\#\{j : j \in \hat{\mathcal{S}}\} + q^{-1}}\right] \leq q$$

where the expectation is taken with respect to ϵ, treating X and $\tilde{X}$ as fixed.

This is not exactly the FDR control we desired. To achieve this type of control, we modify our definition of T: this leads to a slightly more conservative procedure.

Exact FDR control with knockoffs

Let $\tilde{X}$ be a set of knockoffs satisfying (i) and (ii) and let W_j be as defined in (8). Define the estimated support as

$$\hat{\mathcal{S}} = \{j : W_j \geq T\}$$

where

$$T = \min\left\{t \in \mathcal{W} : \frac{1 + \#\{j : W_j \leq -t\}}{\#\max\left(\#\{j : W_j \geq t\}, 1\right)} \leq q\right\}, \quad (T = +\infty \text{ if the set is empty}).$$

Then, for any $q \in [0, 1]$, the knockoff filter procedure satisfies

$$\mathrm{E}\left[\frac{\#\{j : \beta_j^* \neq 0 \cap j \in \hat{\mathcal{S}}\}}{\max(\#\{j : j \in \hat{\mathcal{S}}\}, 1)}\right] \leq q$$

where the expectation is taken with respect to ϵ, treating X and $\tilde{X}$ as fixed.

So why does this work? Without formally proving the results, we will give some intuition. For the β_j^* which are truly zero, i.e., the null hypothesis that $\beta_j^* = 0$ is true, the signs of the corresponding W_j are i.i.d. random variables. Thus,

$$\#\{j : \beta_j^* = 0 \cap W_i \geq t\} \stackrel{d}{=} \#\{j : \beta_j^* = 0 \cap W_i \leq -t\}.$$

To understand the implications of this, we turn to the Figure , taken from Barber and Candès (2015). Here, we see that $\#\{j : \beta_j^* = 0 \cap W_i \geq t\}$ is the number of null points in the shaded region below the diagonal line, whereas $\{j : \beta_j^* = 0 \cap W_i \leq -t\}$ is the number of null points in the shaded region above the diagonal line. For null points, these are evenly distributed above and below the the diagonal line.

Using this idea, we see that the false discovery proportion can be estimated using

$$\frac{\#\{j : \beta_j^* = 0 \cap W_j \geq t\}}{\max(\#\{j : W_j \geq t\}, 1)} \approx \frac{\#\{j : \beta_j^* = 0 \cap W_j \leq -t\}}{\max(\#\{j : W_j \geq t\}, 1)} \leq \frac{\#\{W_j \leq -t\}}{\max(\#\{j : W_j \geq t\}, 1)} = \widehat{\mathrm{FDP}}(t)$$

where $\widehat{\mathrm{FDP}}(t)$ is the knockoff estimate of the false discovery proportion. We can then understand the choice of T as finding a threshold so that

$$T = \min\left\{t \in \mathcal{W} : \widehat{\mathrm{FDP}}(t) \leq q\right\}$$

where we follow the convention that $T = +\infty$ if no such t exists.

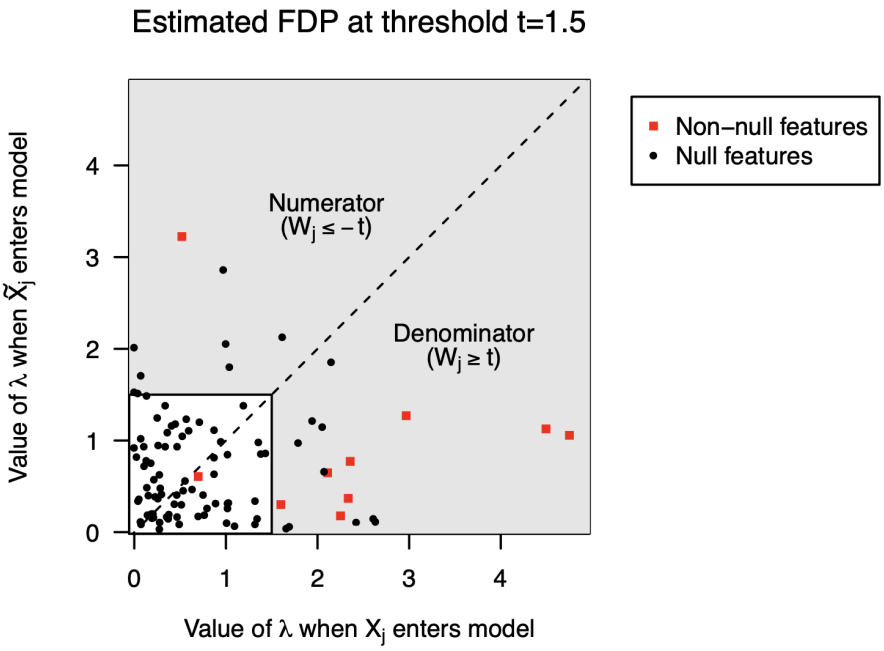

Figure 10: A display of the W_j in a simulated data example with $t = 1.5$. In the figure, the white box represents points with $|W_j| < t$ whereas points above (resp. below) the $y = x$ line and not in the white box have $W_j > t$ (resp. $W_j < -t$).

6.2.2 Constructing knockoffs

We will focus on the simplest case where $n \geq 2p$. Recall that we must construct $\tilde{X}$ where

$$(X, \tilde{X})'(X, \tilde{X}) = \begin{bmatrix} \Sigma & \Sigma - \text{Diag}(a) \\ \Sigma - \text{Diag}(a) & \Sigma \end{bmatrix} = \mathbf{G},$$

where by definition, $\Sigma = X'X$. For $\tilde{X}$ to exist, it must be that $\mathbf{G}$ is positive semidefinite. This is the case if and only if the Schur complement of $\mathbf{G}$:

$$A = \Sigma - (\Sigma - \text{Diag}(a))\Sigma^{-1}(\Sigma - \text{Diag}(a)) = 2\text{Diag}(a) - \text{Diag}(a)\Sigma^{-1}\text{Diag}(a)$$

is positive semidefinite. Note that A is also the Schur complement of the matrix

$$\begin{bmatrix} \Sigma & \text{Diag}(a) \\ \text{Diag}(a) & 2\text{Diag}(a) \end{bmatrix}$$

which is positive semidefinite if and only if

$$\text{Diag}(a) \succeq 0, \quad 2\Sigma - \text{Diag}(a) \succeq 0 \tag{10}$$

where the notation $C \succeq D$ means that $C - D$ is positive semidefinite. Hence, for $\mathbf{G}$ to be positive definite, we need only choose an a so that both of the conditions in (10) are satisfied.

Let $\tilde{U} \in \mathbb{R}^{n \times p}$ be an orthonormal matrix whose column space is orthogonal to that of X, i.e., $U'X = 0$ (which exists since $n \geq 2p$). Since $A \succeq 0$, we can write $A = C'C$ where $C \in \mathbb{R}^{p \times p}$. Then, if we set

$$\tilde{X} = X(I_p - \Sigma^{-1}\text{Diag}\{a\}) + \tilde{U}C,$$

$\tilde{X}$ is a valid knockoff of X.

Exercise. Verify the previous statement.

It only remains to show how to choose an appropriate a. Consider our goal for selecting a: we want an a such that if the jth predictor is actually important (i.e., $\beta_j^* \neq 0$), then we want $\hat{\beta}_j$ to become nonzero for much smaller values of λ than $\hat{\beta}_{j+p}$ (the predictor corresponding to the knockoff $\tilde{X}_j$). For this to happen, we need the correlation between $\tilde{X}_j$ and X_j to be small – i.e., we want $\text{Cor}(\tilde{X}_j, X_j)$ as close to zero as possible. Two approaches we may use are:

1. *Equi-correlated knockoffs:* In this approach, we set $a_j = \min\{2\varphi_p(\Sigma), 1\}$ where $\varphi_p(\Sigma)$ is the smallest eigenvalue of Σ. This way, each pair of predictors and their knockoffs have the same correlation:

$$X_j'\tilde{X}_j = \min\{1 - 2\varphi_p(\Sigma), 1\}.$$

2. *Minimum average correlation knockoffs:* We may also determine a so that the average of the $X_j'\tilde{X}_j$ across $j = 1, \ldots, p$ is minimized, subject to the constraints from (10). This particular a is obtained by solving the convex optimization problem

$$\text{argmin}_{a \in \mathbb{R}^p} \sum_j |1 - a_j| \quad \text{subject to } a_j \geq 0, \text{diag}(a) \succeq 2\Sigma$$

We will not cover how to construct knockoffs in the more complicated setting where $n < p$, let alone $n \ll p$. However, this is an active area of research and developments are relatively well-documented at this URL.

References

Rina Foygel Barber and Emmanuel J. Candès. Controlling the false discovery rate via knockoffs. *Ann. Statist.*, 43(5):2055–2085, 10 2015. doi: 10.1214/15-AOS1337. URL `https://doi.org/10.1214/15-AOS1337`.

Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *Ann. Statist.*, 44(2):813–852, 04 2016. doi: 10.1214/15-AOS1388. URL `https://doi.org/10.1214/15-AOS1388`.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *The Journal of Machine Learning Research*, 15(1):2869–2909, 2014.

Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.