Week 3-1: Lasso regression

Announcement

- HW1 posted
- Lab 3 on Friday (no Zoom) @ midnight

Last time

- Ridge regression

Today

- Lasso regression
- Lab 3 intro

References

- ISL Chapter 6
- James Witten's Zoom notes
- Robert Tibshirani (1996) Regression Shrinkage and Selection via the Lasso, JRSSB 58:267–288
- Rodney Dixon, Trevor Hastie, Jerome Friedman and Robert Tibshirani (2008) Lasso Angle Regression, AOS 32:407–451

Lasso

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\arg\min} \left\{ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

Lasso, short for "least absolute selection operator", which shrinks some coefficients and sets others to 0, hence tries to retain the good properties of both subset selection and ridge regression.

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

where $\text{lasso}(y, \lambda, \text{data}) = \text{glmnet}(y, \text{data}, \alpha = 1)$.

Why the lasso will often produce coefficients that are exactly 0?

Trevor Robert Tibshirani (1996):

Consider $n = 2$

- The ellipse centered on the ballsize $\text{lasso}(y, \text{lambda}=0)$ (lasso, ridge, OLS) and the lasso, ridge, OLS estimator. The contours are centered at $\text{lasso}(y, \text{lambda}=0)$.
- The constraint region for the lasso is the diamond $|\beta_1| + |\beta_2| \le t$, and for ridge it is $\beta_1^2 + \beta_2^2 \le t$.
- The contours of the residual error function for this data are centered at the least squares and the unpenalized estimate (lasso at lambda=0) and are ellipsoidal.
- The solution to the lasso regression is found at $(0, t)$. Ridge will also be a contact point for the contours to hit and hence solutions will rarely be sparse.

Bias and variance

The bias and variance of linear regression estimates behave in the same general manner as they did for ridge regression. The bias increases as $\lambda \uparrow$ (regression of shrinkage increases), and the variance decreases as $\lambda \uparrow$ (amount of shrinkage increases).

Source: Ryan Tibshirani's lecture

Degrees of freedom

Degrees of freedom is a way of quantifying the effective number of parameters used by a fitting procedure.

Linear regression

In linear regression setting, the degrees of freedom of a model is the number of free parameters.

Ridge regression

In ridge regression, we define the effective degrees of freedom as $\text{df}(\text{lambda}) = \text{tr}(X(X^T X + \text{lambda} I_p)^{-1} X^T) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$ (hence, $\text{df}(0) = p$ and $\text{df}(\infty) = 0$), which is a monotone decreasing function of lambda.

- $\text{lambda} = 0$ (OLS) → the effective degrees of freedom is p
- $\text{lambda} \to \infty$ (ridge) → the effective degrees of freedom $\to 0$

Lasso regression

In lasso, the degrees of freedom is the number of nonzero components in $\text{lasso}(y, \text{lambda}, \text{data})$.

How to choose lambda?

Most popular approach is to use cross validation to choose lambda and then apply the one-standard error rule.

- Consider a set of values of lambda: $\text{lambda}_1, \ldots, \text{lambda}_k$ (e.g. $\text{seq}(0, 0.1, \text{length}=100)$)
- For each lambda_i, perform 5 fold cross validation and compute the prediction error $\text{MSE}(\text{lambda}_i)$ (for lasso, $\text{MSE}(\text{lambda}_i) = \text{lasso.cv}(y, \text{data}, \text{lambda}=\text{lambda}_i)\$\text{cvm}$).
- Find the tuning parameter lambda^* that minimizes $\text{MSE}(\text{lambda}_i)$. Define the one-standard-error rule: $\text{lambda}_{\text{1se}} = \text{lambda}_{\text{cv.min}} + \text{lambda}_{\text{cv.1se}} = \text{lambda}_{\text{cv.min}} + (\text{lambda}_{\text{cv.min}} - \text{lambda}_{\text{cv.1se}})$
- We fit ridge regression with $\text{lambda}_{\text{1se}}$, since it gives us the simplest model that is within one standard error of the best model. Simpler models are more interpretable and often comparable predictive performance by using the one-standard error rule.
  - Compute the cross validation standard error: $\text{cvsd}(\text{lambda}_i) = \text{sqrt}(\text{var}(\text{lambda}_i)/5)$ (for lasso, $\text{cvsd}(\text{lambda}_i) = \text{lasso.cv}(y, \text{data}, \text{lambda}=\text{lambda}_i)\$\text{cvsd}$)
  - Choose the largest lambda such that $\text{lambda}_i \le \text{lambda}_{\text{cv.min}} + \text{cvsd}(\text{lambda}_{\text{cv.min}})$ (for lasso, $\text{lambda}_i \le \text{lambda}_{\text{cv.min}} + \text{cvsd}(\text{lambda}_{\text{cv.min}})$)

Suppose $n = 10$, 10 truly large coefficients, 90 zero coefficients. The one-standard error rule chooses a model with 10 variables, whereas the usual rule chooses a model with 20.

Orthogonal design

Under the orthogonal design $X^T X = I_p$, lasso estimation can be simplified as solving $\text{lasso}(y, \text{lambda}) = \text{soft}(\text{lm.fit}(X, y)\$\text{coefficients}, \text{lambda})$ (since $\text{lm.fit}(X, y)\$\text{coefficients} = X^T y$).

Then the lasso estimates is given by $\hat{\beta}_j^{\text{lasso}} = \text{S}(\text{lm.fit}(X, y)\$\text{coefficients}, \lambda)$, **lambda** where $\text{S}(\text{lambda}, \text{lambda}) = \text{soft}(\text{lambda}, \text{lambda}) = 0$, $\text{lambda} \to \text{lambda} - \text{lambda}$ for large lambda.

It performs the so called soft thresholding:

Wavelet denoising

Soft thresholding is commonly used for orthonormal bases.

- Suppose that we observe $y = \text{lambda} + \epsilon$ (like a time series)
- Let Φ be an $n \times n$ orthonormal basis, $\Phi^T \Phi = \Phi \Phi^T = I_n$ (for example, a matrix of discrete wavelet lambda lambda)
- Let $\theta = \Phi^T \text{lambda}$ and $y = \Phi^T y$ (hence $y = \Phi \theta + \epsilon$)

Now to recovering lambda from y, $\hat{\text{lambda}} = \Phi \hat{\theta}$ where $\hat{\theta} = \text{soft}(y, \lambda)$ ($\hat{\theta} = \text{lasso}(y, \lambda)$).

```
library(tidy) ## for
library(glmnet) ## for lasso
library(ggplot2) ## for ggplot
library(gridExtra) ## for grid.arrange

## Exercise Chapter 6:lasso wavelet denoising
## 1 dim smooth signal - Fourier basis (orthogonal design)
lambda = sin(seq(0,4*pi,length=2^8))
```

```
## lasso
Phi = matrix(rnorm(256*256), nrow=256)
Phi = qr.Q(qr.decomp(Phi, tol=1e-10, complete=TRUE))
t(Phi) %*% Phi - diag(256)
Phi[1:5,1:5]; Phi[1:5,256-4:0]
Phi[1:5,1:5]; Phi[1:5,256-4:0]
Phi[1:5,1:5]; Phi[1:5,256-4:0]

## Check that Phi is orthonormal
t(Phi) %*% Phi - diag(256)
Phi[1:5,1:5]; Phi[1:5,256-4:0]
Phi[1:5,1:5]; Phi[1:5,256-4:0]
Phi[1:5,1:5]; Phi[1:5,256-4:0]

## Check that lambda is smooth
lambda[1:5]; lambda[256-4:0]

## Observe lambda
y = Phi %*% lambda
y[1:5]; y[256-4:0]

## Define soft function
soft = function(lambda, lambda) {
  ifelse(abs(lambda) > lambda, lambda - lambda*sign(lambda), 0)
}

## Check
lambda = soft(y, lambda)
lambda[1:5]; lambda[256-4:0]

## Define lasso_lambda function
lasso_lambda = function(y, lambda) {
  n = length(y)
  Phi = qr.Q(qr.decomp(Phi, tol=1e-10, complete=TRUE))
  theta = t(Phi) %*% y
  theta = soft(theta, lambda)
  return(Phi %*% theta)
}

## Define lasso_boot function
lasso_boot = function(y, lambda) {
  n = length(y)
  ind = sample(1:n, n, replace=TRUE)
  return(lasso_lambda(y[ind], lambda[ind]))
}

## Define lasso_boot_boot function
lasso_boot_boot = function(y, lambda) {
  n = length(y)
  ind = sample(1:n, n, replace=TRUE)
  return(lasso_lambda(y[ind], lambda[ind]))
}

## Check
lambda.lasso = lasso_lambda(y, lambda)
lambda.lasso[1:5]; lambda.lasso[256-4:0]

## Check lasso_boot function
lambda.lasso.boot = lasso_boot(y, lambda)
lambda.lasso.boot[1:5]; lambda.lasso.boot[256-4:0]

## Check lasso_boot_boot function
lambda.lasso.boot.boot = lasso_boot_boot(y, lambda)
lambda.lasso.boot.boot[1:5]; lambda.lasso.boot.boot[256-4:0]

## Check
lambda.lasso.boot.boot[1:5]; lambda.lasso.boot.boot[256-4:0]

## Define lasso_boot_boot_n function
lasso_boot_boot_n = function(y, lambda, n) {
  replicate(n, lasso_boot_boot(y, lambda))
}

## Check
lambda.lasso.boot.boot.100 = lasso_boot_boot_n(y, lambda, 100)
lambda.lasso.boot.boot.100[1:5,1:5]; lambda.lasso.boot.boot.100[256-4:0,1:5]

## Define lasso_boot_boot_n_df function
lasso_boot_boot_n_df = function(y, lambda, n) {
  return(rowSums(lasso_boot_boot_n(y, lambda, n) != 0))
}

## Check
lambda.lasso.boot.boot.100.df = lasso_boot_boot_n_df(y, lambda, 100)
lambda.lasso.boot.boot.100.df[1:5]; lambda.lasso.boot.boot.100.df[256-4:0]

## Define lasso_boot_boot_n_df_df function
lasso_boot_boot_n_df_df = function(y, lambda, n) {
  return(data.frame(df = lasso_boot_boot_n_df(y, lambda, n), lambda = lambda))
}

## Check
lambda.lasso.boot.boot.100.df.df = lasso_boot_boot_n_df_df(y, lambda, 100)
lambda.lasso.boot.boot.100.df.df[1:5,]; lambda.lasso.boot.boot.100.df.df[256-4:0,]

## Define lasso_boot_boot_n_df_df_ggplot function
lasso_boot_boot_n_df_df_ggplot = function(y, lambda, n, ...) {
  return(ggplot(lasso_boot_boot_n_df_df(y, lambda, n), aes(x=lambda, y=df), ...) + geom_point())
}

## Check
lambda.lasso.boot.boot.100.df.df.ggplot = lasso_boot_boot_n_df_df_ggplot(y, lambda, 100)
lambda.lasso.boot.boot.100.df.df.ggplot
```

```
## lasso_boot_boot_n_df_df_ggplot_lambda function
lasso_boot_boot_n_df_df_ggplot_lambda = function(y, lambda, n, ...) {
  return(ggplot(lasso_boot_boot_n_df_df(y, lambda, n), aes(x=lambda, y=df), ...) + geom_point() + geom_line(aes(x=lambda, y=lambda), ...))
}

## Check
lambda.lasso.boot.boot.100.df.df.ggplot.lambda = lasso_boot_boot_n_df_df_ggplot_lambda(y, lambda, 100)
lambda.lasso.boot.boot.100.df.df.ggplot.lambda
```



```
## lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange function
lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange = function(y, lambda, n, ...) {
  return(grid.arrange(lasso_boot_boot_n_df_df_ggplot(y, lambda, n, ...), lasso_boot_boot_n_df_df_ggplot_lambda(y, lambda, n, ...), ...))
}

## Check
lambda.lasso.boot.boot.100.df.df.ggplot.lambda.grid.arrange = lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange(y, lambda, 100)
lambda.lasso.boot.boot.100.df.df.ggplot.lambda.grid.arrange
```



```
## lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange_ggsave function
lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange_ggsave = function(y, lambda, n, filename, ...) {
  return(ggsave(filename, lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange(y, lambda, n, ...), ...))
}

## Check
lambda.lasso.boot.boot.100.df.df.ggplot.lambda.grid.arrange.ggsave = lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange_ggsave(y, lambda, 100, "./lasso_boot_boot_n_df_df_ggplot_lambda_grid.arrange.pdf")
```

Wavelet reconstruction

Can reconstruct the response by $\text{lambda} = \Phi \theta$ or $\Phi \hat{\theta}$ (lasso). The objective is $\text{RSS}(\text{lambda}) = \|\epsilon\|_2^2 = \|y - \Phi \theta\|_2^2 = \|\Phi \hat{\theta} - \Phi \theta\|_2^2 = \|\hat{\theta} - \theta\|_2^2 = \text{RSS}(\theta)$

- The ℓ_2^2 penalty, Parseval Lemma, $\|\cdot\|_2$ is reducing sparsity.

```
## plot DCT/DWT coefficients
par(mfrow=c(2,2)); plot(lambda.lasso.boot.boot.100.df.df.ggplot.lambda.grid.arrange$data$df, pch=19, cex=0.5)
par(mfrow=c(2,2))
## Error: 1 dim 256 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 2 dim 65536 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 3 dim 16777216 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 4 dim 4294967296 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 5 dim 1099511627776 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 6 dim 281474976710656 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 7 dim 72057594037927936 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 8 dim 18446744073709551616 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 9 dim 4722366482869645213696 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 10 dim 1208925819614629174706176 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 11 dim 309485009821345068724781056 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 12 dim 79228162514264337593543950336 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 13 dim 20282409603651670423947251286016 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 14 dim 5192296858534827628530496329220096 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 15 dim 1329227995784915872903807060280344576 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
##
## Error: 16 dim 340282366920938463463374607431768211456 obs
## 1.0 ---------------------------------
##
## 0.5 ---------------------------------
##
## 0.0 ---------------------------------
##
## -0.5 ---------------------------------
##
## -1.0 ---------------------------------
##
##     0   1   2   3   4   5   6   7   8
## NULL
## NULL
## NULL
```