Statistics 452: Statistical Learning and Prediction

Chapter 6, Part 2: Shrinkage Methods

Brad McNeney

2017-10-07

Shrinkage Methods

- Fit a model that contains all p predictors using a method that shrinks the coefficient estimates towards zero.
- This biases the estimates, but reduces variance.
- We will discuss two shrinkage methods, ridge regression and the lasso.

Ridge Regression

- Penalize the criterion function, RSS, to favour smaller coefficient values.
- The ridge regression criterion function is

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

  where $\lambda \geq 0$ is a tuning parameter.
- The ridge regression estimator $\hat{\beta}_\lambda^R$ is the minimizer of this criterion.
- Note: The penalty is on the criterion used to fit the model, **not** on the measure of fit (compare with C_p).
- The penalty term has two components, the tuning parameter and the sum of squared coefficients.

Tuning Parameter, λ

- We do **not** penalize the intercept.
- $\lambda = 0$ gives least squares
- $\lambda > 0$ will lead to estimates of $\beta_1, \ldots, \beta_p$ that are "shrunken" towards zero
- To be practical, we need a method for choosing the tuning parameter.

SS Coefficients, $\sum_{j=1}^{p} \beta_j^2$

- $\sum_{j=1}^{p} \beta_j^2$ is the square of the length of the vector $(\beta_1, \ldots, \beta_p)$ of non-intercept coefficients.
- The length is the Euclidean or ℓ_2 norm of the vector.
  - Sometimes ridge regression is called ℓ_2-penalized regression.

Scaling Predictors

- The least squares solution is said to be scale invariant.
  - If we multiply a predictor X_j by a constant c, the least squares solution $\hat{\beta}_j$ is multiplied by $1/c$ so that $X_j\hat{\beta}_j$ doesn't change.
- The same is not true for ridge regression.
  - $X_j\beta^R_{\lambda,j}$ depends on the scale of X_j; e.g., on the units X_j is measured in.
- We typically standardize each predictor by subtracting its mean and dividing by its sample SD.
  - Then the units of each X_j don't matter.

Application to Credit Data

```r
uu <- url("http://www-bcf.usc.edu/~gareth/ISL/Credit.csv")
Credit <- read.csv(uu,row.names=1)
head(Credit,n=3)
```

```
##    Income Limit Rating Cards Age Education Gender Student Married
## 1  14.891  3606    283     2  34        11   Male      No     Yes
## 2 106.025  6645    483     3  82        15 Female     Yes     Yes
## 3 104.593  7075    514     4  71        11   Male      No      No
##   Ethnicity Balance
## 1 Caucasian     333
## 2     Asian     903
## 3     Asian     580
```

Least Squares for Comparison

- Set up the design matrix and response ourselves and pass to the `lm.fit()` function, which does the fitting for `lm()`.

```
Xfull <- model.matrix(Balance ~ ., data=Credit)
head(Xfull,n=3)
```

```
##   (Intercept)  Income Limit Rating Cards Age Education GenderFemale
## 1           1  14.891  3606    283     2  34        11            0
## 2           1 106.025  6645    483     3  82        15            1
## 3           1 104.593  7075    514     4  71        11            0
##   StudentYes MarriedYes EthnicityAsian EthnicityCaucasian
## 1          0          1              0                  1
## 2          1          1              1                  0
## 3          0          0              1                  0
```

```
Y <- Credit$Balance
```

```r
# Standardize predictors
predInds <- 2:ncol(Xfull)
Xfull[,predInds] <- scale(Xfull[,predInds])
Y <- Credit$Balance
lsfit <- lm.fit(Xfull,Y)
lsfit$coefficients
```

```
##        (Intercept)             Income              Limit
##         520.015000        -275.014651         440.650711
##             Rating              Cards                Age
##         175.848092          24.305139         -10.589809
##          Education       GenderFemale         StudentYes
##          -3.434150          -5.330027         127.884163
##         MarriedYes     EthnicityAsian EthnicityCaucasian
##          -4.162747           7.333463           5.059778
```

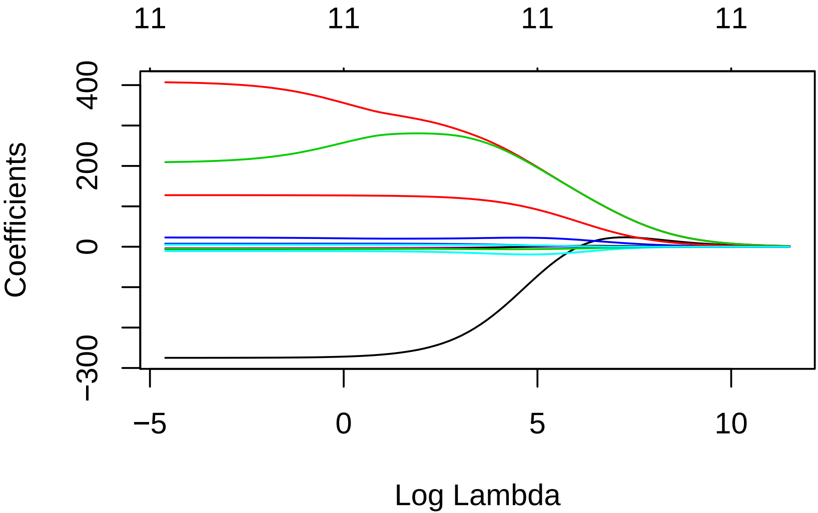
Ridge Regression

- Find the ridge regression solution for each λ on a grid.

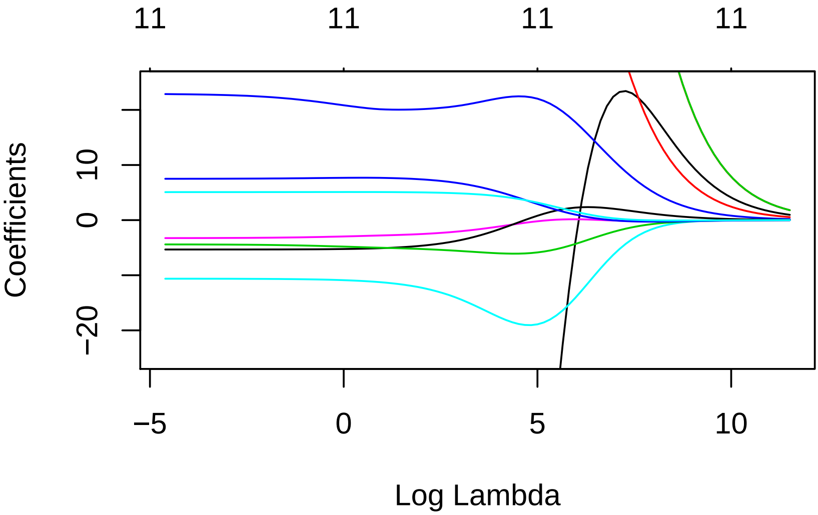
```
library(glmnet) # install.packages("glmnet"), if necessary
Xfull <- Xfull[,-1] # glmnet will add its own intercept
lambdas <- 10^{seq(from=-2,to=5,length=100)}
rrfit <- glmnet(Xfull,Y,alpha=0,lambda=lambdas)
round(cbind(coef(rrfit,s=lambdas[1]),coef(rrfit,s=lambdas[50])),4)
```

```
## 12 x 2 sparse Matrix of class "dgCMatrix"
##                           1         1
## (Intercept)         520.0150  520.0150
## Income             -274.9070 -202.0820
## Limit               407.0633  276.0537
## Rating              209.3416  266.0201
## Cards                22.8687   21.2405
## Age                 -10.6288  -15.4669
## Education            -3.2623   -1.7511
## GenderFemale         -5.3312   -3.0400
## StudentYes          127.6673  117.6251
## MarriedYes           -4.4077   -5.7662
## EthnicityAsian        7.4978    6.2090
## EthnicityCaucasian    5.0805    4.7190
```

```r
plot(rrfit,xvar="lambda")
```

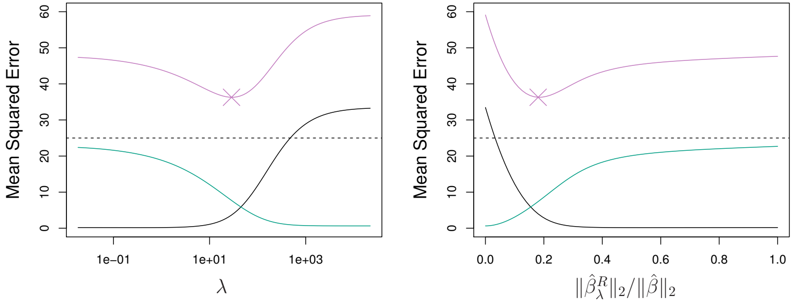


```r
plot(rrfit,xvar="lambda",ylim=c(-25,25))
```



Bias-Variance Tradeoff

- Recall that the MSE is the variance plus bias squared
- The least squares estimate of the regression coefficients is unbiased and therefore so are the predictions $X\hat{\beta}$.
- Penalizing introduces bias into the predictions, but reduces variance.
- Illustrated in the text on a simulated dataset.



- Figure 6.5 of the text. The MSE is in purple, variance in green and squared bias in black.
  - Minimum MSE is at λ of about 30.

The Lasso

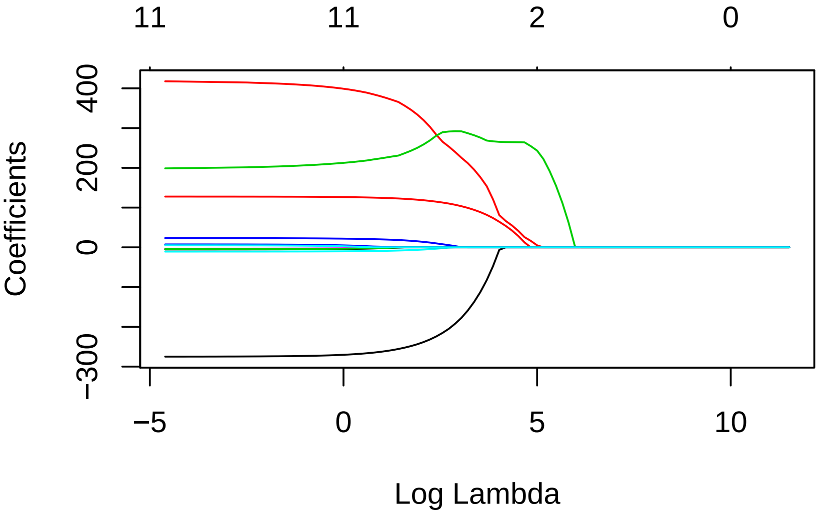
- A drawback of ridge regression is that it does not *select* a subset of predictors.
  - The final model includes all p coefficients, shrunken toward zero.
  - Not good for interpretation.
- An alternative called the lasso does model selection and shrinkage.
- The lasso replaces the ℓ_2 penalty of ridge regression with an ℓ_1 penalty; i.e., the lasso estimator $\hat{\beta}^L$ minimizes the criterion

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$

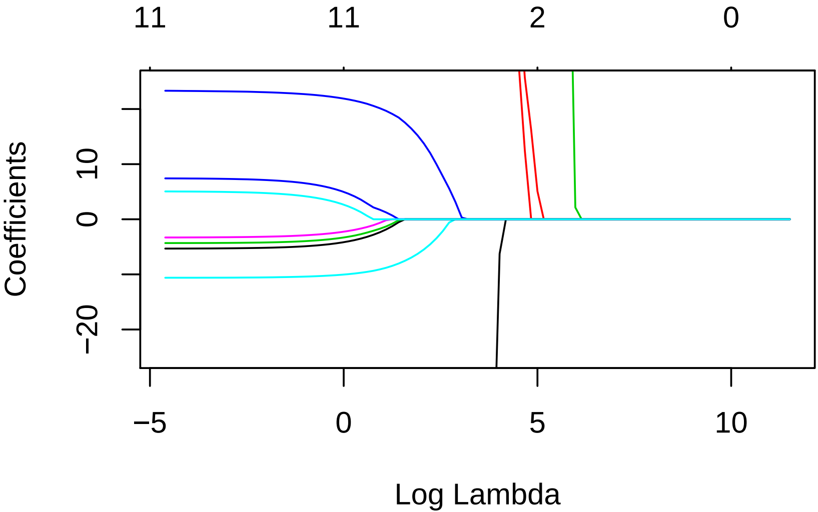
- It turns out that the lasso can shrink estimates to zero, and hence de-select predictors.
  - Variable-selected models are said to be sparse.

The Lasso on the Credit Data

```
lafit <- glmnet(Xfull,Y,alpha=1,lambda=lambdas)
plot(lafit,xvar="lambda")
```



```r
plot(lafit,xvar="lambda",ylim=c(-25,25))
```



- After $\log \lambda > 6$ or so all coefficients have been set to zero.

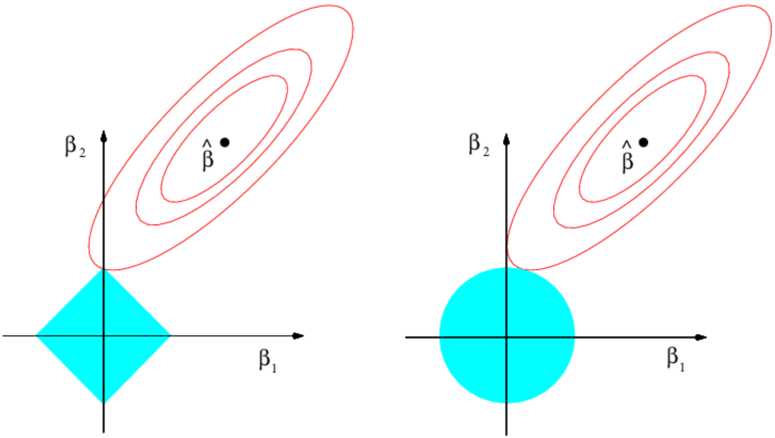
Equivalent Representation of Ridge and Lasso

- One can show that for a given λ there is an s such that the lasso solution $\hat{\beta}_\lambda^L$ is the solution to the constrained minimization of RSS subject to

$$\sum_{j=1}^{p} |\beta_j| \leq s$$

- Similarly, the ridge regression solution $\hat{\beta}_\lambda^R$ is the solution to the constrained minimization of RSS subject to

$$\sum_{j=1}^{p} \beta_j^2 \leq s$$



- Figure 6.7 of the text. The shaded regions are where the constraints are satisfied for the lasso (left) and ridge regression (right). The contours are of the RSS. The lasso solution zeroes out the β_1 coefficient.

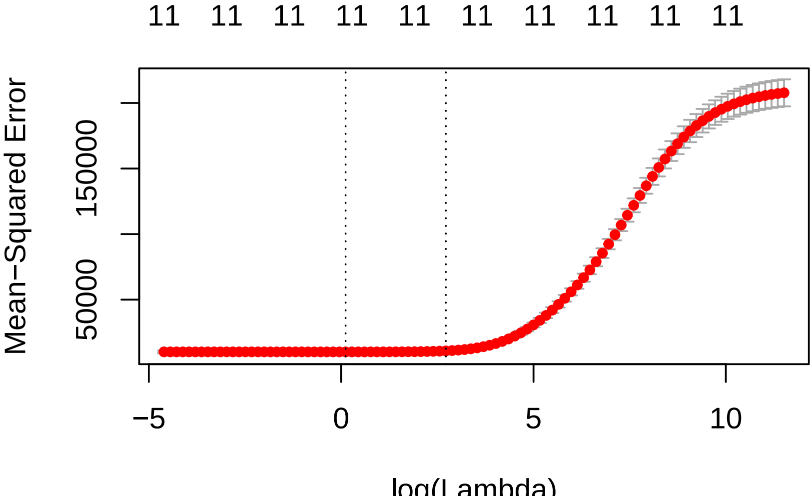
Selecting the Tuning Parameter

- We can select the λ that minimizes estimated test set error over a grid of λ values.
  - Estimate test set error by cross-validation.
- Then fit the model with this best λ.
- Convenience function `cv.glmnet()` will do most of the work for us.

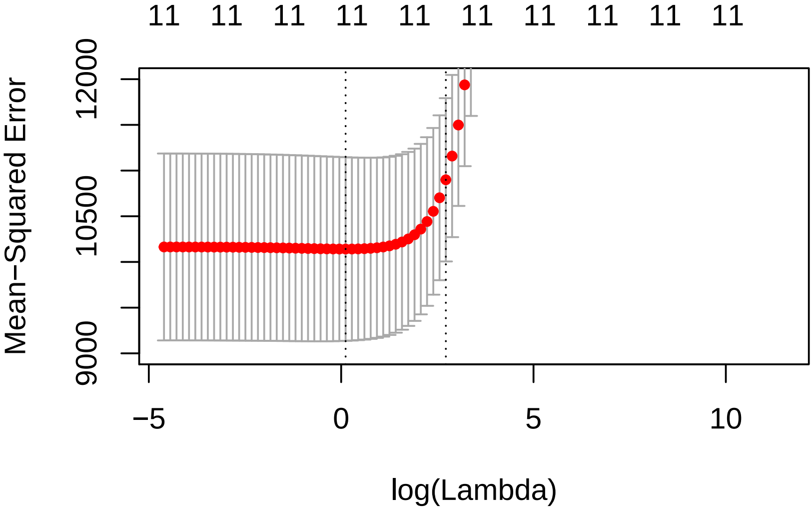
Credit Data Example

- First ridge regression

```r
# Ridge regression
cv.rrfit <- cv.glmnet(Xfull,Y,alpha=0,lambda=lambdas)
plot(cv.rrfit)
```



```r
plot(cv.rrfit,ylim=c(9000,12000))
```



Error Bars

- The error bars are $\pm$ on SD of the MSE estimates across the ten folds.
- Hastie & co. (ESL, page 216) advocate the "one-standard-error" rule: Use the most parsimonious model whose error is no more than one SD above the error of the best model.
- Acknowledges that the MSEs are only estimates.
  - Rather *ad hoc* rule though.
- In this example, the best model **is** the most parsimonious.

Fitted Model with Best λ

```r
rr.best.lam <- cv.rrfit$lambda.min
rr.best.lam
```

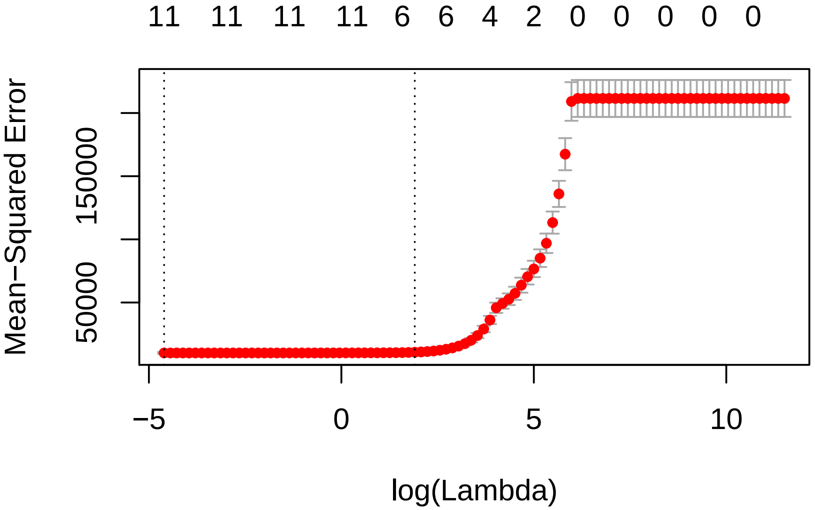
```
## [1] 1.123324
```

```r
rr.best <- glmnet(Xfull,Y,alpha=0,lambda=rr.best.lam)
coef(rr.best)
```

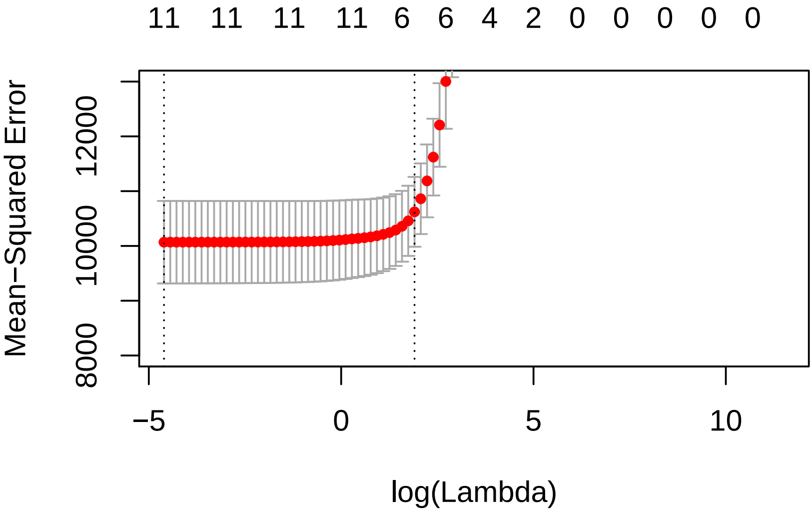
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)        520.015000
## Income            -271.804820
## Limit              383.876803
## Rating             229.524804
## Cards               22.039961
## Age                -10.882706
## Education           -3.100901
## GenderFemale        -5.232930
## StudentYes         127.136235
## MarriedYes          -4.606307
## EthnicityAsian       7.535274
## EthnicityCaucasian   5.080022
```

- Now lasso

```r
cv.lafit <- cv.glmnet(Xfull,Y,alpha=1,lambda=lambdas)
plot(cv.lafit)
```



```r
plot(cv.lafit,ylim=c(8000,13000))
```



- Shouldn't we choose $\lambda = 0.01$ here too?

```r
la.best.lam <- cv.lafit$lambda.min
la.best.lam
```

```
## [1] 0.01
```

```r
la.best <- glmnet(Xfull,Y,alpha=0,lambda=la.best.lam)
coef(la.best)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                             s0
## (Intercept)         520.015000
## Income             -275.064049
## Limit               472.635954
## Rating              143.903043
## Cards                25.675726
## Age                 -10.556898
## Education            -3.596967
## GenderFemale         -5.327198
## StudentYes          128.083888
## MarriedYes           -3.930132
## EthnicityAsian        7.175776
## EthnicityCaucasian    5.039896
```

Summary of Credit Data

- Shrinkage estimation did little.
- We'd probably just use the least-squares estimates and SEs:
  - Though we know model selection methods found a model with better MSE than the full model (see week 6 exercises).

```
lsfit <- lm(Balance ~ ., data=Credit)
round(summary(lsfit)$coefficients,3)
```

```
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -479.208     35.774 -13.395    0.000
## Income               -7.803      0.234 -33.314    0.000
## Limit                 0.191      0.033   5.824    0.000
## Rating                1.137      0.491   2.315    0.021
## Cards                17.724      4.341   4.083    0.000
## Age                  -0.614      0.294  -2.088    0.037
## Education            -1.099      1.598  -0.688    0.492
## GenderFemale        -10.653      9.914  -1.075    0.283
## StudentYes          425.747     16.723  25.459    0.000
## MarriedYes           -8.534     10.363  -0.824    0.411
## EthnicityAsian       16.804     14.119   1.190    0.235
## EthnicityCaucasian   10.107     12.210   0.828    0.408
```