Penalized Estimation and Shrinkage

Challenge:

- Training set error (training MSE or training classification error) **always goes down** if we add more explanatory variables, higher degree polynomial terms, interactions, ...
- Test set error goes down for a while, then back up.

Two Running Examples

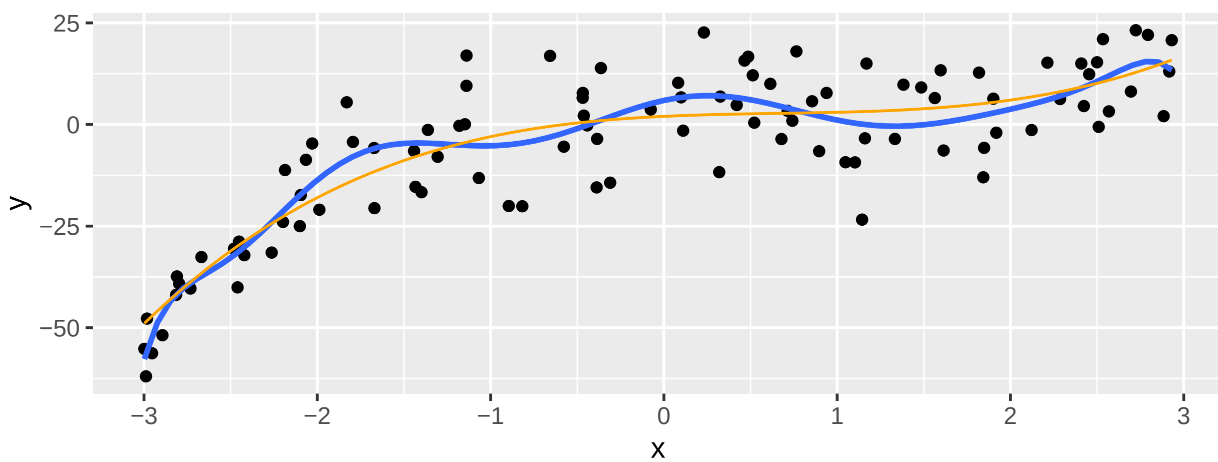
Example 1: Polynomial Regression

Let's consider fitting polynomials of degree 10 to data generated from the following model:

$$
\begin{aligned}
X_i &\sim \text{Uniform}(-3, 3) \\
Y_i &= 2 + 2X_i - 2X_i^2 + X_i^3 + \varepsilon_i \\
\varepsilon_i &\sim \text{Normal}(0, 20) \\
i &= 1, \ldots, 100
\end{aligned}
$$

```r
head(train_data, 3)
```

```
##           x         y
## 1 2.4542876 12.363632
## 2 2.1219489 -1.370620
## 3 0.8551751  5.703446
```




```r
lm_fit <- lm(y ~ poly(x, 10), data = train_data)
coef(lm_fit)
```

```
##   (Intercept)  poly(x, 10)1  poly(x, 10)2  poly(x, 10)3  poly(x, 10)4
##     -5.327173    143.973446    -70.584437     57.095163     -1.945988
##  poly(x, 10)5  poly(x, 10)6  poly(x, 10)7  poly(x, 10)8  poly(x, 10)9
##      9.234072    -10.096321    -13.167566      1.011708     11.278393
## poly(x, 10)10
##    -12.077299
```

What's the problem?

Those coefficient estimates are too big!!

Example 2: Linear Regression with $p = 10$ possible explanatory variables, only 3 relevant

$$\begin{aligned} X_{i1}, \ldots, X_{i10} &\sim \text{Uniform}(-3, 3) \\ Y_i &= 0 + 1X_{i1} + 2X_{i2} + 3X_{i3} + 0X_{i4} + 0X_{i5} + 0X_{i6} + 0X_{i7} + 0X_{i8} + 0X_{i9} + 0X_{i10} + \varepsilon_i \\ \varepsilon_i &\sim \text{Normal}(0, 20) \\ i &= 1, \ldots, 100 \end{aligned}$$

```r
head(train_data)
```

```
##            X1         X2         X3         X4         X5          X6
## 1 -1.01938653 -2.0985918 -2.3551104 -2.4098541 -1.8176376  1.18202520
## 2 -0.04593549 -2.8209434 -1.5988486 -0.3659999  1.6435141 -2.12081627
## 3 -2.47423902 -2.3711388 -1.0917102 -2.6031636 -0.8780756  0.04128208
## 4 -0.52725392  0.2402836 -2.8992715 -0.7564033 -2.2546738  1.42963905
## 5 -1.14103982  0.6296736  0.2408957 -1.1774389 -2.7576011  0.92439794
## 6  0.48656256  0.8602587  0.2032003 -1.0902985  1.5552264  0.40651091
##           X7         X8         X9        X10          y
## 1 -0.1388858 -1.3060173  0.3686385  0.3196567 -16.596887
## 2 -2.6157885 -2.3707292 -0.7515223 -0.8928924 -41.070053
## 3 -1.4040666 -0.7556161 -1.3073824  1.1664008   8.287668
## 4 -1.0899567 -0.8544949  1.0751195  2.5880689 -12.897716
## 5  2.1341929  2.3695607  0.5087685 -0.2380945   4.501106
## 6  1.0186687  0.5036288  1.2631464 -1.3265513  49.773204
```

```r
lm_fit <- lm(y ~ ., data = train_data)
coef(lm_fit)
```

```
## (Intercept)          X1          X2          X3          X4          X5
##   3.3172496   2.4199628   3.2782373   2.9279032   1.3142178  -1.3582212
##          X6          X7          X8          X9         X10
##  -0.6889279   1.5074253  -0.2138175   0.4920307   0.8193166
```

What's the problem?

(Most of) those coefficient estimates are too big!!

Solution: Shrinkage, a.k.a. Penalized Estimation, a.k.a. Regularized Estimation

Application to polynomial regression example

Recall that bias and variance are about behavior of estimators across all possible training sets. To understand behavior, we will look at estimates from each method across 100 randomly generated training sets (each of size $n = 100$) from the model

$$\begin{aligned}
X_i &\sim \text{Uniform}(-3, 3) \\
Y_i &= 2 + 2X_i - 2X_i^2 + X_i^3 + \varepsilon_i \\
\varepsilon_i &\sim \text{Normal}(0, 20)
\end{aligned}$$

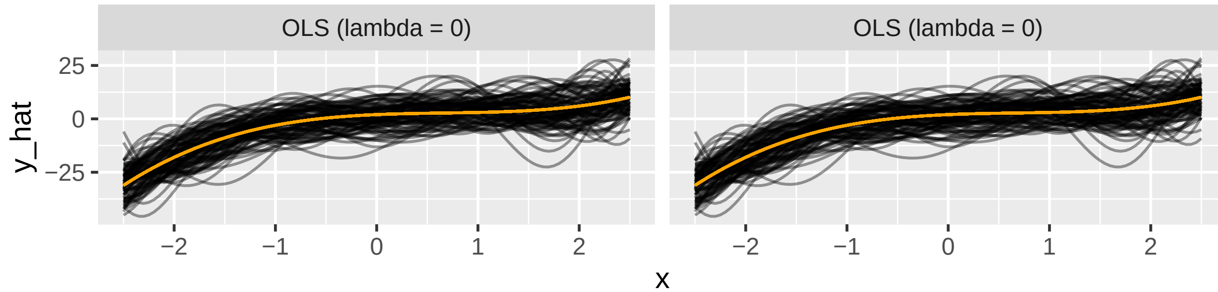
Code suppressed to avoid distraction, but here is what I did:

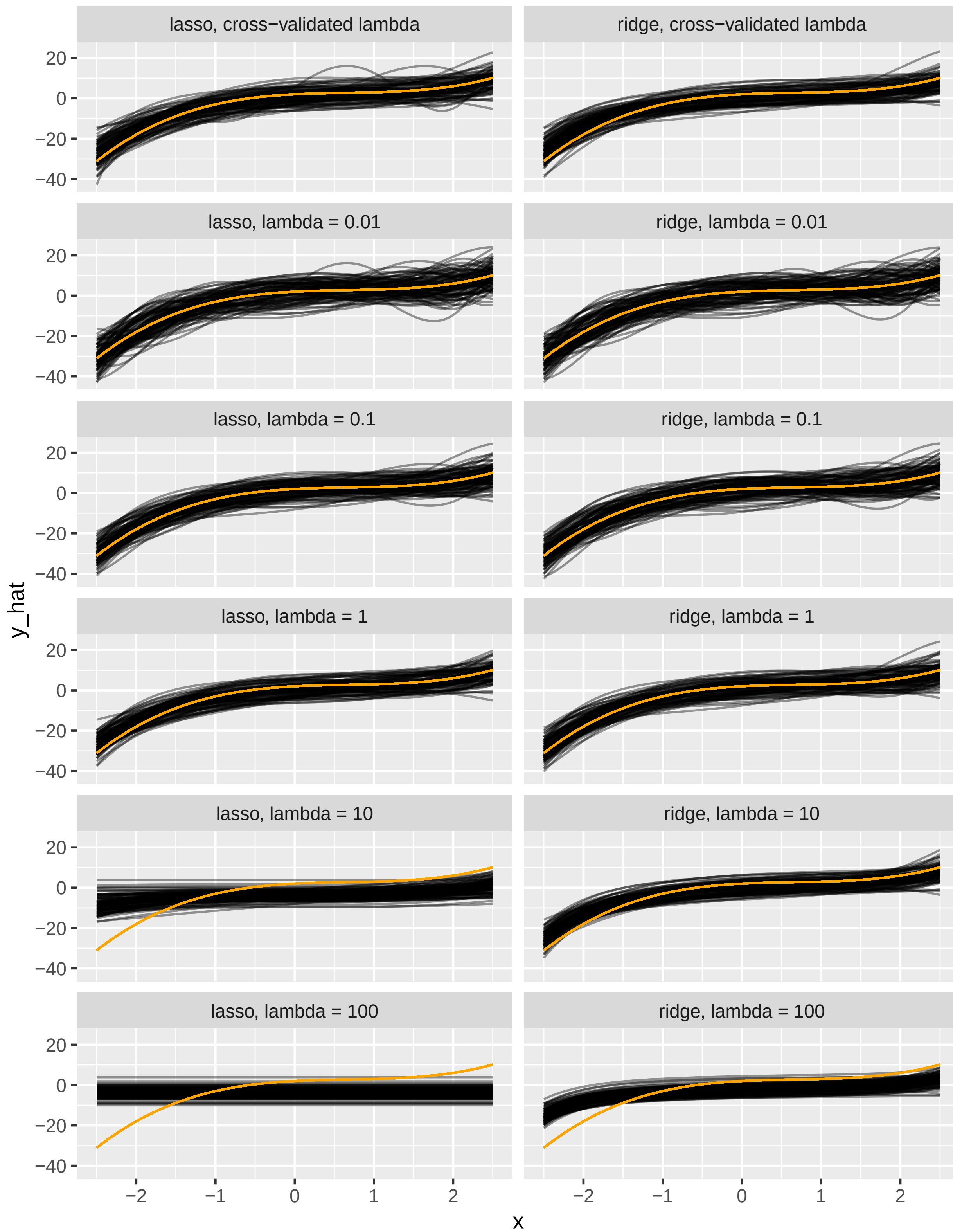
- Simulate 100 different data sets of size 100 from this model
- For each simulated data set, fit a degree 10 polynomial using each of the following procedures:
  - OLS ($\lambda = 0$)
  - LASSO with the following choices of λ:
    - 0.01, 0.1, 1, 10, 100
    - Cross-validated choice of λ (the selected value was generally a little less than 1)
  - Ridge Regression with the following choices of λ:
    - 0.01, 0.1, 1, 10, 100
    - Cross-validated choice of λ (the selected value was generally a little less than 1)

Plot the resulting estimates $\hat{f}(x)$, with the true function $f(x)$ overlaid in orange.

```
## Warning in bind_rows_(x, .id): binding factor and character vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```


OLS (lambda = 0)
OLS (lambda = 0)
y_hat
25
0
−25
−2
−1
0
1
2
x


lasso, cross−validated lambda
ridge, cross−validated lambda
lasso, lambda = 0.01
ridge, lambda = 0.01
lasso, lambda = 0.1
ridge, lambda = 0.1
lasso, lambda = 1
ridge, lambda = 1
lasso, lambda = 10
ridge, lambda = 10
lasso, lambda = 100
ridge, lambda = 100
y_hat
20
0
−20
−40
−2
−1
0
1
2
x

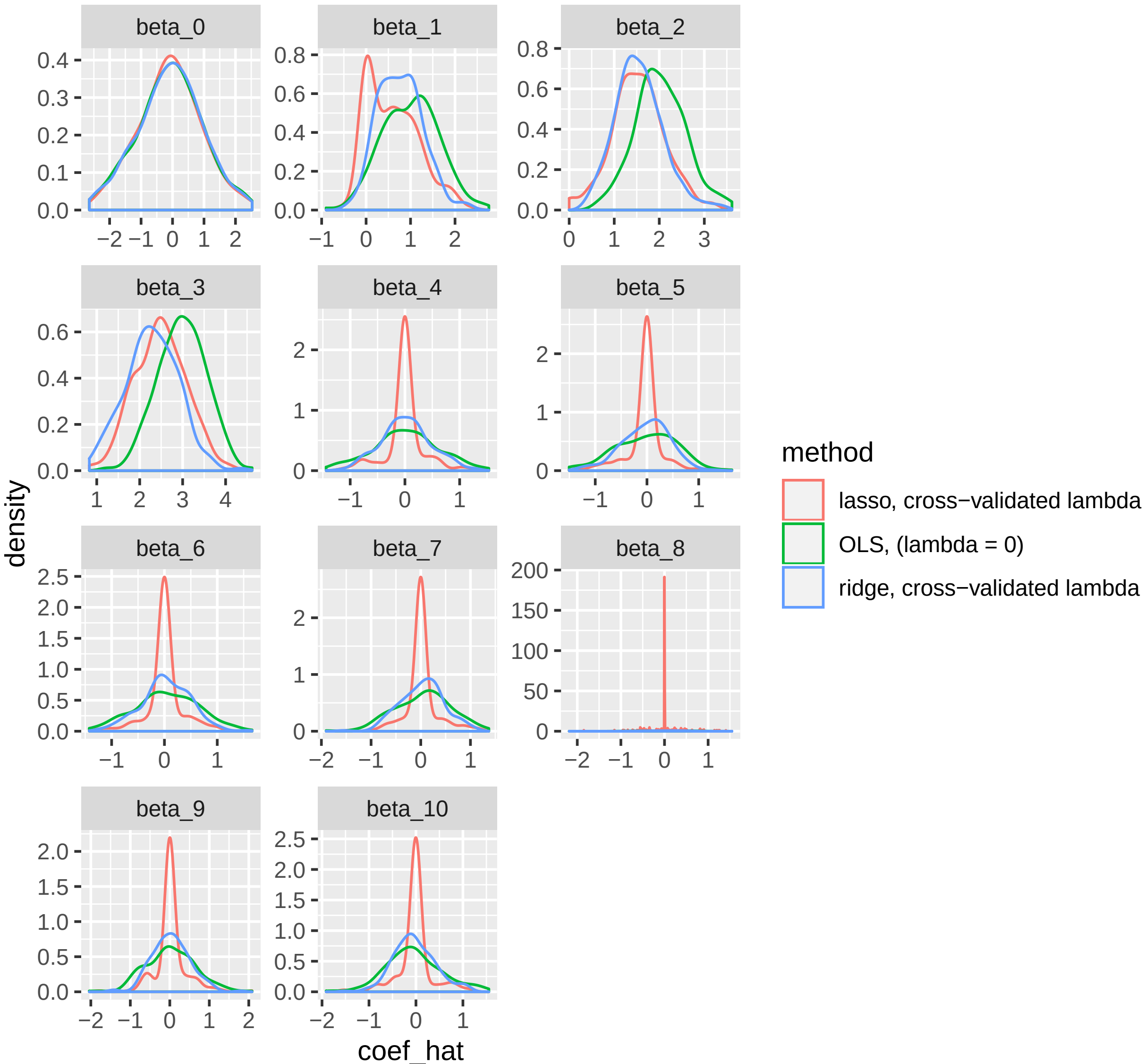
Example with large p

I simulated 200 data sets (each with $n = 100$ observations) from the model with $p = 10$ explanatory variables:

$$
\begin{aligned}
X_{i1}, \dots, X_{i10} &\sim \text{Uniform}(-3, 3) \\
Y_i &= 0 + 1X_{i1} + 2X_{i2} + 3X_{i3} + 0X_{i4} + 0X_{i5} + 0X_{i6} + 0X_{i7} + 0X_{i8} + 0X_{i9} + 0X_{i10} + \varepsilon_i \\
\varepsilon_i &\sim \text{Normal}(0, 20) \\
i &= 1, \dots, 100
\end{aligned}
$$

For each simulated data set, I estimated the model with OLS, Ridge regression, and LASSO.

Here are density plots summarizing the resulting coefficient estimates.



- We introduced a small amount of bias in estimates of the three non-zero coefficients, in exchange for a large reduction in variance of estimates of the remaining coefficients.
- This bias-variance trade-off in coefficient estimates translates into a bias-variance trade-off in prediction errors.
- LASSO is more aggressive in setting coefficient estimates to 0 than Ridge regression
  - LASSO is preferred if many β's are close to 0
  - Ridge is preferred if not

Example: Prostrate Cancer

This example comes from Chapter 3 of Elements of Statistical Learning. Here's a quote from that book describing the setting:

> The data for this example come from a study by Stamey et al. (1989). They examined the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. The variables are log cancer volume (lcavol), log prostate weight (lweight), age, log of the amount of benign prostatic hyperplasia (lbph), seminal vesicle invasion (svi), log of capsular penetration (lcp), Gleason score (gleason), and percent of Gleason scores 4 or 5 (pgg45).

Our goal is to understand the relationship between these explanatory variables and the level of prostrate-specific antigen.

```r
library(readr)
library(dplyr)
library(ggplot2)
library(caret)
library(leaps) # functionality for best subsets regression

prostrate <- read_table2("http://www.evanlray.com/data/ESL/prostate.data")
prostrate <- prostrate %>%
  select(-ID, -train)
head(prostrate)
```

```
## # A tibble: 6 x 9
##   lcavol lweight   age  lbph   svi   lcp gleason pgg45   lpsa
##    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>
## 1 -0.580    2.77    50 -1.39     0 -1.39       6     0 -0.431
## 2 -0.994    3.32    58 -1.39     0 -1.39       6     0 -0.163
## 3 -0.511    2.69    74 -1.39     0 -1.39       7    20 -0.163
## 4 -1.20     3.28    58 -1.39     0 -1.39       6     0 -0.163
## 5  0.751    3.43    62 -1.39     0 -1.39       6     0  0.372
## 6 -1.05     3.23    50 -1.39     0 -1.39       6     0  0.765
```

```r
dim(prostrate)
```

```
## [1] 97  9
```

```r
set.seed(76471)

# perform train/test split
train_test_split_inds <- caret::createDataPartition(prostrate$lpsa)

prostrate_train <- prostrate %>% slice(train_test_split_inds[[1]])
prostrate_test <- prostrate %>% slice(-train_test_split_inds[[1]])
```

Ordinary least squares

```r
# ordinary least squares
lm_fit <- train(
  form = lpsa ~ .,
  data = prostrate_train,
  method = "lm", # method for fit
  trControl = trainControl(method = "none")
)

coef(lm_fit$finalModel)
```

```
##  (Intercept)       lcavol      lweight          age         lbph
##  1.074261455  0.542229797  0.645149313 -0.021409418  0.042629191
##          svi          lcp      gleason        pgg45
##  0.256741436  0.141087559 -0.034505914 -0.002417642
```

```r
mean((prostrate_test$lpsa - predict(lm_fit, newdata = prostrate_test))^2)
```

```
## [1] 0.6559828
```

LASSO

We will ask the train function in the caret package to do cross-validation for us to select the penalty parameter λ.

To do this, we specify:

- `trainControl(method = "CV")`
- `tuneGrid` is a data frame with values of parameters to tune. In this case:
  - `alpha = 1` says we want the LASSO penalty.
  - `lambda` is a grid of possible values for the penalty λ between 0 and 1.

```
lasso_fit <- train(
  form = lpsa ~ .,
  data = prostrate_train,
  method = "glmnet", # method for fit
  trControl = trainControl(method = "cv"),
  tuneGrid = data.frame(
    alpha = 1,
    lambda = seq(from = 0, to = 1, length = 100)
  )
)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

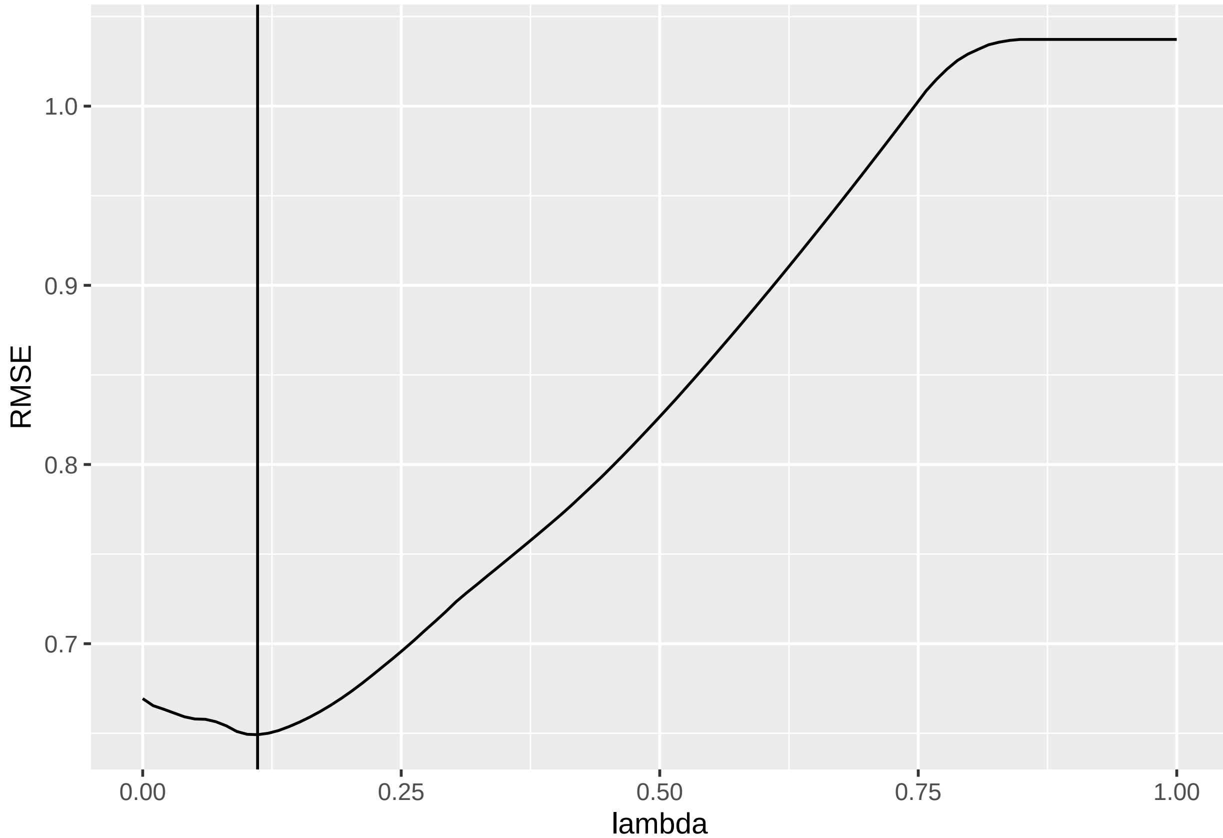
```
# results from cross-validation
head(lasso_fit$results)
```

```
##   alpha     lambda      RMSE  Rsquared       MAE    RMSESD RsquaredSD
## 1     1 0.00000000 0.6693608 0.6560105 0.5445037 0.2386452  0.2385461
## 2     1 0.01010101 0.6654135 0.6629152 0.5433890 0.2390742  0.2297848
## 3     1 0.02020202 0.6634566 0.6671238 0.5443160 0.2404571  0.2225922
## 4     1 0.03030303 0.6613107 0.6724992 0.5452711 0.2423820  0.2182043
## 5     1 0.04040404 0.6592097 0.6780962 0.5448129 0.2456140  0.2170136
## 6     1 0.05050505 0.6579912 0.6832044 0.5434143 0.2488389  0.2185099
##       MAESD
## 1 0.1889924
## 2 0.1924833
## 3 0.1955039
## 4 0.1989230
## 5 0.2035812
## 6 0.2083963
```

```
lasso_fit$bestTune$lambda
```

```
## [1] 0.1111111
```

```
ggplot(data = lasso_fit$results, mapping = aes(x = lambda, y = RMSE)) +
  geom_line() +
  geom_vline(xintercept = lasso_fit$bestTune$lambda)
```




```r
# coefficient estimates with the selected value of lambda
coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.60521184
## lcavol      0.44551065
## lweight     0.36064396
## age         .
## lbph        .
## svi         0.15258791
## lcp         0.09389823
## gleason     .
## pgg45       .
```

```r
# test set MSE
mean((prostrate_test$lpsa - predict(lasso_fit, newdata = prostrate_test))^2)
```

```
## [1] 0.6762453
```

Ridge Regression

We will ask the train function in the caret package to do cross-validation for us.

To do this, we specify:

- `trainControl(method = "CV")`
- `tuneGrid` is a data frame with values of parameters to tune. In this case:
    - `alpha = 0` says we want the ridge penalty.
    - `lambda` is a grid of possible values for the penalty λ between 0 and 1.

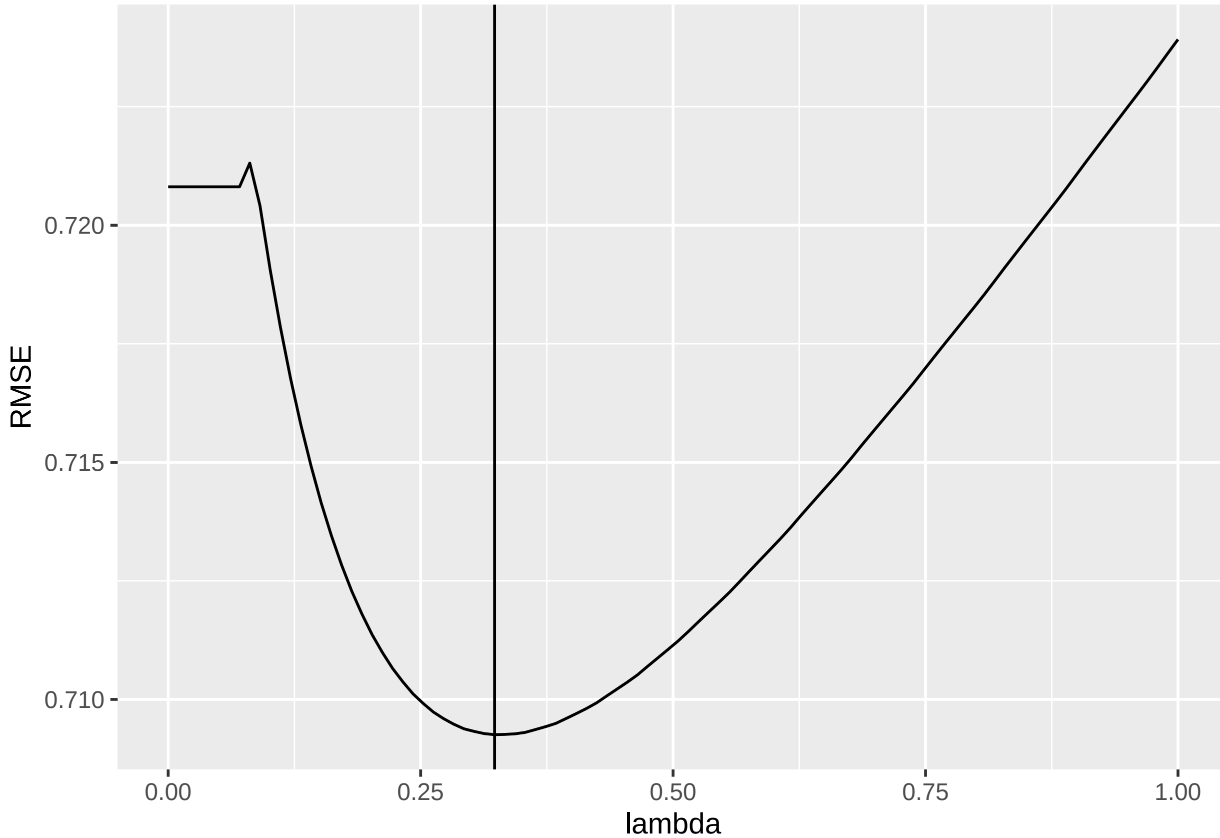
```
ridge_fit <- train(
  form = lpsa ~ .,
  data = prostrate_train,
  method = "glmnet", # method for fit
  trControl = trainControl(method = "cv"),
  tuneGrid = data.frame(
    alpha = 0,
    lambda = seq(from = 0, to = 1, length = 100)
  )
)
# results from cross-validation
head(ridge_fit$results)
```

```
##   alpha     lambda      RMSE  Rsquared     MAE    RMSESD RsquaredSD
## 1     0 0.00000000 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
## 2     0 0.01010101 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
## 3     0 0.02020202 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
## 4     0 0.03030303 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
## 5     0 0.04040404 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
## 6     0 0.05050505 0.7208097 0.6565662 0.58741 0.2759379  0.3306834
##       MAESD
## 1 0.2282589
## 2 0.2282589
## 3 0.2282589
## 4 0.2282589
## 5 0.2282589
## 6 0.2282589
```

```
ridge_fit$bestTune$lambda
```

```
## [1] 0.3232323
```

```
ggplot(data = ridge_fit$results, mapping = aes(x = lambda, y = RMSE)) +
  geom_line() +
  geom_vline(xintercept = ridge_fit$bestTune$lambda)
```




```r
# coefficient estimates with the selected value of lambda
coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  0.5700103955
## lcavol       0.3700351459
## lweight      0.4916160434
## age         -0.0077902053
## lbph         0.0419606882
## svi          0.3702317985
## lcp          0.1374004240
## gleason      0.0224245412
## pgg45       -0.0008791053
```

```r
# test set MSE
mean((prostrate_test$lpsa - predict(ridge_fit, newdata = prostrate_test))^2)
```

```
## [1] 0.635933
```