STK2100: Solutions Week 6

Lars H. B. Olsen

18.02.2021

ISLR book

Exercise 6.2

a)

Alternative *iii* is the correct choice; The lasso limits the number of predictors, thus it reduces the inherent variance at the cost of an increase in bias. This can be interpreted as follows. Removing a predictor from the model is equivalent to saying the removed feature does not have a strong relationship with the target value, this may be a biased statement but it decreases the variance as a lower number of values need to be estimated from data.

b)

Alternative *iii* is the correct choice; Ridge regression will produce more biased models as it shrinks predictors that don't have as a strong relationship with the target variable; the variance will decrease at the cost of an increase in bias.

c)

Alternative *ii* is the correct choice; Nonlinear methods have a higher variance than regular least scares. The curve will follow the observations more tightly than otherwise, causing the model to perform better when there is a underlying non-linear relationship between the predictors and the target variable.

Exercise 6.3

a)

Alternative *iv* is the correct choice; There will be a monotonically decrease in *training RSS* as a bigger set of solutions becomes feasible and an increase in variance occurs.

b)

Alternative *ii* is the correct choice; An increase in s means there will be an increase in flexibility in the model. For $s = 0$, we have the intercept model, so the *test RSS* will be high. As s increase we get a more flexible model which are able to fit the data better and we get a decrease in test RSS. However, when s is too big, we start to overfit the training data and we are not able to exploit the bias-variance-trade-off, hence, the test RSS starts to increase.

c)

Alternative *iii* is the correct choice; When s increase we get a more flexible model, hence, the variance will steadily increase.

d)

Alternative *iv* is the correct choice; When s increase we go from the intercept model ($s = 0$) with a high squared bias to a more flexible model with a lower squared bias. Unbiased in the limit. Hence, the squared bias will steadily decrease.

e)

Alternative v is the correct choice; As it names states, the irreducible error will remain constant. The irreducible error is independent of model parameters and thus independent of s.

Exercise 6.10

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

a)

We generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model $Y = X\beta + \epsilon$, where β has some elements that are exactly equal to zero.

If we generate the data in another way, we get quite different results. And the seed highly influence the results.

```
# Seet seed for reproducibility
set.seed(2021)

# Set the parameters
p = 20
n = 1000

# Create a matrix of n \times p
X = matrix(rnorm(n*p), n, p)

# Create a vector of beta and randomly assign 0 to some betas
b = rnorm(p)
b[1] = b[3] = b[5] = b[7] = b[9] = 0

# Generating the error terms
error = rnorm(n)

# Multiplying X and beta to get y
y = X %*% b + error
```

b)

Split your data set into a training set containing 100 observations and a test set containing 900 observations.

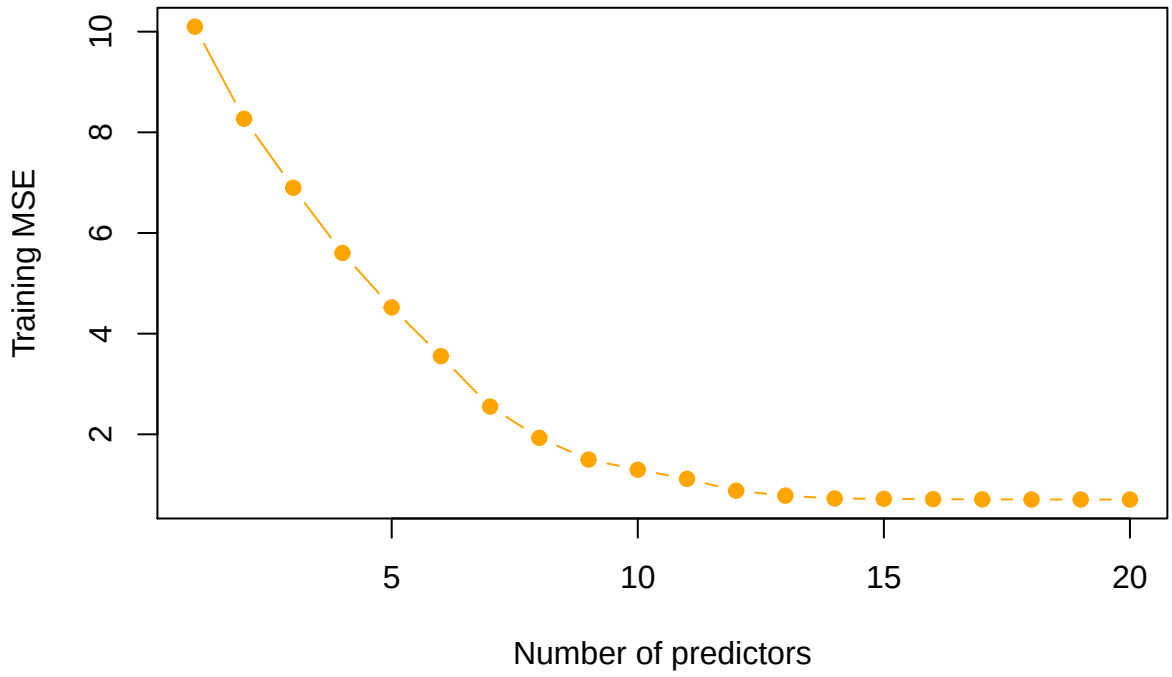
```
# Create a vector train of length 100 and assign them random
# index number chosen from 1 to n.
train = sample(seq(n), 100, replace=FALSE)

# Create a vector test which contains rest of the indices
test = (-train)

# Subset X matrix for the given sequence in train and test
```

```
x.train = X[train,]
x.test = X[test,]
y.train = y[train]
y.test = y[test]
```

c)

Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

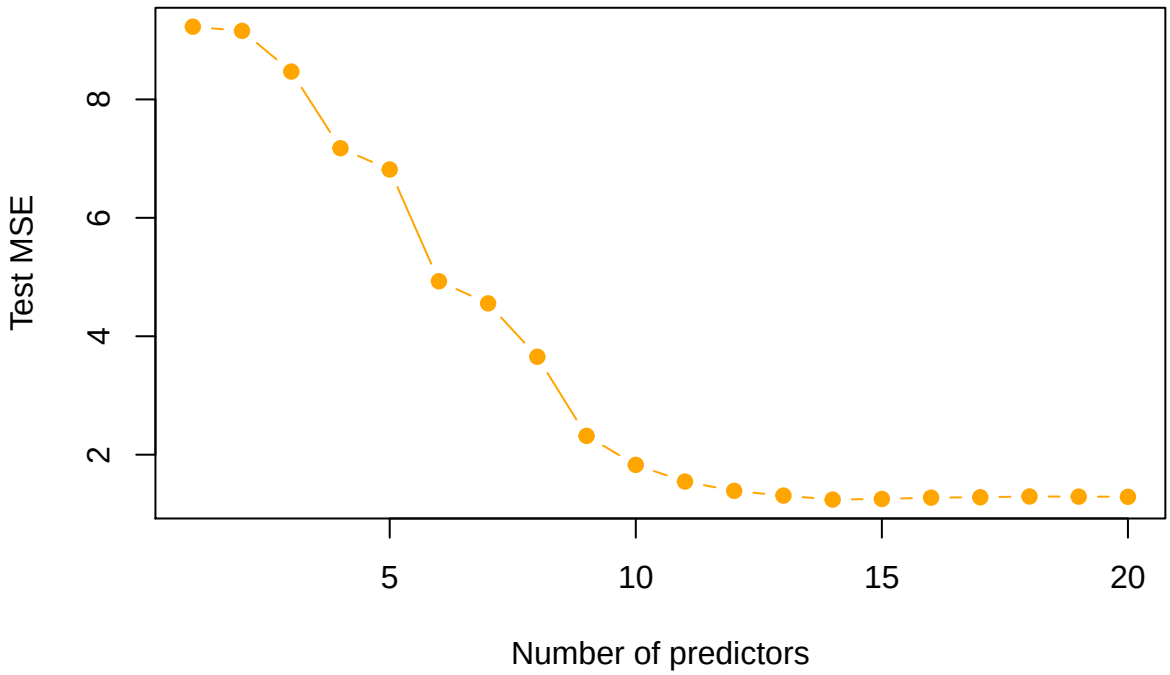
```
# Import the leaps library to conduct the best subset selection
library(leaps)

# Combine training data and training response into a common data frame
data.train = data.frame(y = y.train, x = x.train)

# See ?regsubsets. Conducts model selection by exhaustive search.
regfit.full = regsubsets(y ~ ., data = data.train, nvmax = p)

# Construct design matrix
train.mat = model.matrix(y ~ ., data = data.train, nvmax = p)

# Array to store the training MSE
val.errors = rep(NA, p)

# Iterate over the number of predictors
for (i in 1:p) {
    # Get the coefficients of the model with i predictors
    coefi = coef(regfit.full, id = i)

    # Compute the predictions for the training data
    pred = train.mat[, names(coefi)] %*% coefi

    # Compute the corresponding training MSE
    val.errors[i] = mean((pred - y.train)^2)
}

# Plot the training MSE
plot(val.errors, xlab = "Number of predictors",
     ylab = "Training MSE", pch = 19, type = "b",col="orange")
```



d)

Plot the test set MSE associated with the best model of each size.

```
# Create a data frame of the test data.
data.test = data.frame(y = y.test, x = x.test)

# Create the corresponding design matrix
test.mat = model.matrix(y ~ ., data = data.test, nvmax = p)

# Array to store the test MSE
val.errors = rep(NA, p)

# Iterate over the p models
for (i in 1:p) {
  # Get the coefficients of the ith model
  coefi = coef(regfit.full, id = i)

  # Compute the predictions of the test instances
  pred = test.mat[, names(coefi)] %*% coefi

  # Comptue the test MSE
  val.errors[i] = mean((pred - y.test)^2)
}

# Plot the test MSE
plot(val.errors, xlab = "Number of predictors",
     ylab = "Test MSE", pch = 19, type = "b",col="orange")
```



e)

For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

```
which.min(val.errors) # 14
```

```
## [1] 14
```

We get that the model with 14 variables is the one with the lowest test set MSE.

f)

How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.

```
coef(regfit.full, which.min(val.errors))
```

```
##  (Intercept)          x.2          x.4          x.6          x.8         x.10
##  0.007858244 -0.966883088  0.488157768 -0.895973419 -1.224552344 -1.128006294
##         x.12         x.13         x.14         x.15         x.16         x.17
##  0.727396529 -1.020616370 -2.492471074 -0.493006806 -0.852328431  0.352594406
##         x.18         x.19         x.20
##  0.259739532  1.009661083 -0.613034766
```

We zeroed certain parameters in true model. More precisely, we set $\beta_1 = \beta_3 = \beta_5 = \beta_7 = \beta_9 = 0$, and the best subset model, for which the test MSE is minimum, is able to identify those variables and remove them from the model (in addition to removing β_{11}, and note that our model had $\beta_0 = 0$).

g)

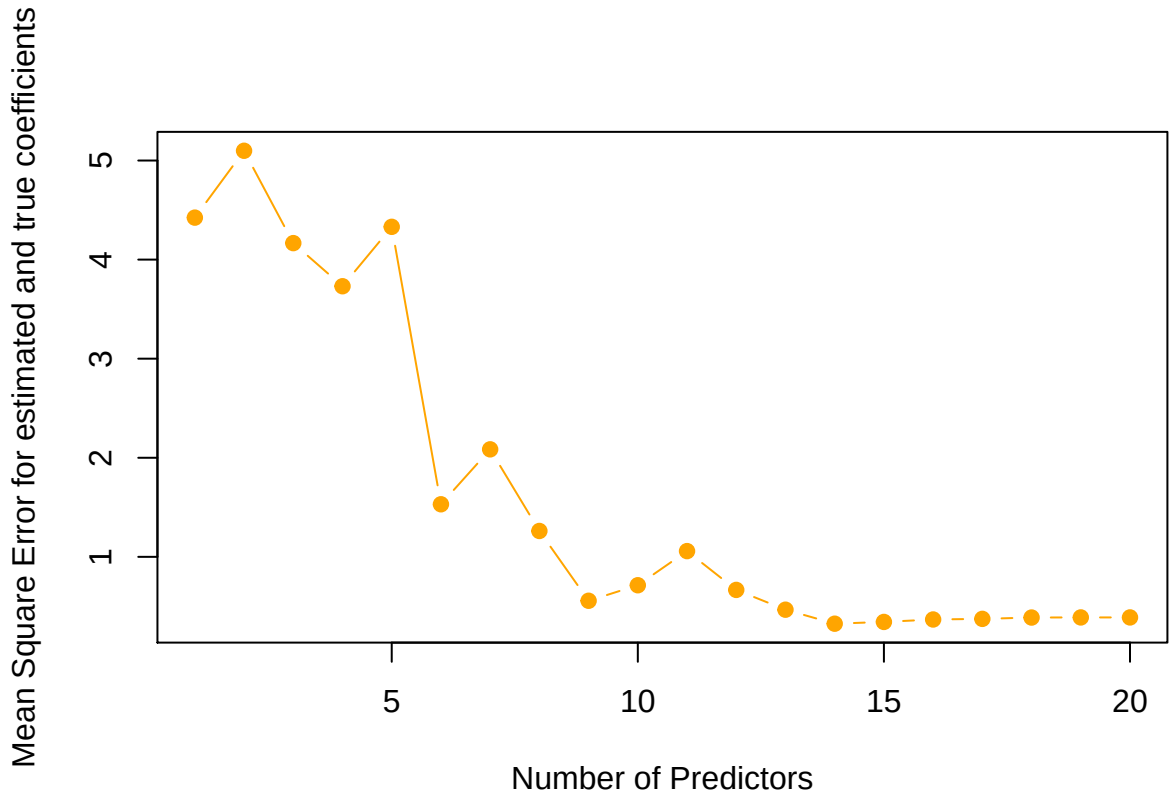
Create a plot displaying $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values of r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the

test MSE plot from (d)?

```
# Array to store the results
val.errors = rep(NA, p)

# Get the colnames of the X matrix
x_cols = colnames(X, do.NULL = FALSE, prefix = "x.")

# Iterate over the different best models
for (i in 1:p) {
  # Get the coefficients of the ith model
  coefi = coef(regfit.full, id = i)

  # Compute the values of interest
  val.errors[i] =
    sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2)
         + sum(b[!(x_cols %in% names(coefi))])^2)
}
which.min(val.errors)
```

```
## [1] 14
```

```
# Plot the values
plot(val.errors, xlab = "Number of Predictors",
     ylab = "Mean Square Error for estimated and true coefficients",
     pch = 19, type = "b", col="orange")
```



It can be seen that the error is minimized for 14 variables. The test MSE is also minimum for the 14 variable model.

We can be lead to the conclude that the model which gives parameter estimates closest to true parameter

estimate also gives the least test MSE, but if we alter the seed above, then we would see that the model which gives parameter estimates closest to true parameter estimate does not need to be the same model as the one that gives the least test MSE, that is, it need not be the best model to predict the response of new observations.

Try to alter the seed yourself and look at how the results change!

Textbook

Exercise 4.3

Consider the parameter h of local regression. Why is it not estimated by a standard method such as maximum likelihood?

If h had been fitted based on the maximum likelihood principle, we would have ended up with an overfitted model. I.e., the model would follow the structure of the training data too much and would not be able to generalize well to an independent test set. The procedure would favor a small h such that the local regression model would be extremely flexible and fit the training data precisely, and thereby increase the likelihood function. Hence, the model would not follow the general tendencies of the data, as would be the case for a larger h, see Figure 4.1 in the textbook.

Exercise 4.5

We are asked to show that $f(x) = \sum_{j=1}^{K+4} \beta_j h_j(x)$ (see (4.10) in the textbook) satisfies three conditions, which characterize cubic splines. Here, $h_j(x) = x^{j-1}$, for $j = 1, 2, 3, 4$, and $h_{j+4}(x) = (x - \xi_j)_+^3$, for $j = 1, 2, \ldots, K$. Note that $a_+ = \max(0, a)$ and $\xi_0 = -\infty$ and $\xi_{K+1} = \infty$. For more intuition, see e.g. Figure 4.5 in the textbook, and section 7.4 in ISLR.

We will here demonstrate it for $K = 1$. This procedure can then easily be generalized to higher values of knots K, but we get a little bit more intricate expressions.

We then have that

$$\begin{aligned} f(x) &= \beta_1 h_1(x) + \beta_2 h_2(x) + \beta_3 h_3(x) + \beta_4 h_4(x) + \beta_5 h_5(x) \\ &= \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 (x - \xi_1)_+^3. \end{aligned}$$

1)

Show that f is a cubic function in each subinterval $[\xi_j, \xi_{j+1})$, for $j = 1, \ldots, K-1$. The function f is obviously a cubic function in the subinterval $[\xi_0, \xi_1) = (-\infty, \xi_1)$ by definition.

In our setting, we will only show that it holds in the interval $[\xi_1, \xi_2) = [\xi_1, \infty)$, as we only consider $K = 1$. In this subinterval, f takes the following shape

$$\begin{aligned} f(x) &= \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 (x - \xi_1)_+^3 \\ &= \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 x^3 - 3\beta_5 x^2 \xi_1 + 3\beta_5 x \xi_1^2 - \beta_5 \xi_1^3 \\ &= (\beta_1 - \beta_5 \xi_1^3) + (\beta_2 + 3\beta_5 \xi_1^2) x + (\beta_3 - 3\beta_5 \xi_1) x^2 + (\beta_4 + \beta_5) x^3, \end{aligned}$$

which is clearly a cubic function.

2)

Show that f has two continuous derivatives. Thsi statement is obviously true on each subinterval, so we only have to consider the what happens with f at $x = \xi_1$, i.e., on the knot.

For $x \uparrow \xi_1$, we have that

$$f''_-(x) = 2\beta_3 + 6\beta_4 x.$$

While for $x \downarrow \xi_1$, we have that

$$f''_+(x) = 2(\beta_3 - 3\beta_5\xi_1) + 6(\beta_4 + \beta_5)x = 2\beta_3 + 6\beta_4 x + 6(\beta_5 x - \beta_5\xi_1).$$

Here I have used '-' and '+' in the subscript of f to indicate that $x < \xi_1$ and $\xi_1 \leq x$, respectively.

If we evaluate $f''_-(x)$ and $f''_+(x)$ in $x = \xi_1$ (limit-wise), we get

$$\lim_{x\uparrow\xi_1} f''_-(x) = 2\beta_3 + 6\beta_4\xi_1$$

and

$$\lim_{x\downarrow\xi_1} f''_+(x) = 2\beta_3 + 6\beta_4\xi_1.$$

Thus, f has two continuous derivatives.

3)

If we differentiate the functions in *b)* one more time, we get that $f'''_-(x) = 6\beta_4$ and $f'''_+(x) = 6\beta_4 + 6\beta_5$. That is, we get that f has a jump in the third derivative at the knot point ξ_1. So the third derivative is what we call a step function.

All of the calculations above can easily be extended to the case with K knots, it only involves more terms, but the ideas are exactly the same.

Exercise 4.6

We are asked to prove (4.12). We want to find the $\hat{\theta}$ which minimses $D = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega\theta$, where Ω is a symmetric matrix. We follow the same strategy as when we computed the regular least square solutions. I.e., first, expand D,

$$\begin{aligned} D &= (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega\theta \\ &= y^Ty - y^TN\theta - \theta^TN^Ty + \theta^TN^TN\theta + \lambda\theta^T\Omega\theta \\ &= y^Ty - (N^Ty)^T\theta - \theta^T(N^Ty) + \theta^T(N^TN)\theta + \lambda\theta^T\Omega\theta, \end{aligned}$$

then differentiate it with respect to θ, (see (69) and (81) in http://www2.imm.dtu.dk/pubdb/edoc/imm3274.pdf),

$$\begin{aligned} D' &= -N^Ty - N^Ty + [N^TN + (N^TN)^T]\theta + \lambda(\Omega + \Omega^T)\theta \\ &= -2N^Ty + [N^TN + N^TN]\theta + \lambda(\Omega + \Omega)\theta \\ &= -2N^Ty + 2N^TN\theta + 2\lambda\Omega\theta, \end{aligned}$$

set it equal to zero and solve for θ,

$$\begin{aligned} -2N^Ty + 2N^TN\theta + 2\lambda\Omega\theta &= 0 \\ 2N^TN\theta + 2\lambda\Omega\theta &= 2N^Ty \\ N^TN\theta + \lambda\Omega\theta &= N^Ty \\ (N^TN + \lambda\Omega)\theta &= N^Ty \\ \theta &= (N^TN + \lambda\Omega)^{-1}N^Ty. \end{aligned}$$

Thus, the value $\hat{\theta}$ which minimses D is $\hat{\theta} = (N^TN + \lambda\Omega)^{-1}N^Ty$, which is what we were asked to prove.