Homework2

Nik Krieger

2018-02-26

library(MASS)  
library(boot)  
library(ISLR)  
library(glmnet)  
library(pls)  
library(leaps)

# Week 4 Exercises:

## Chapter 5 - Exercise 9

#### Task (a)

The estimate for the population mean of medv is the sample mean of medv:

mean(Boston$medv)

## [1] 22.53281

### Task (b)

The estimate for the population mean of medv is the sample standard deviation of medv divided by the square root of the sample size of medv.

sd(Boston$medv) / sqrt(length(Boston$medv))

## [1] 0.4088611

This means that our estimate of the population mean of medv will differ from the true mean by an average of $408.86.

### Task (c)

Here is a bootstrap procedure to estimate the standard error of the population mean estimate (i.e., the sample mean) of medv.

mean.function = function(data.vector, indices)  
 return(mean(data.vector[indices]))  
  
set.seed(0)  
boot(data = Boston$medv, statistic = mean.function, R=1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = mean.function, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 22.53281 0.0005490119 0.3982063

The bootstrap procedure yielded a standard error of 0.3982063, which is only slightly less than the conventionally calculated value of 0.4088611.

### Task (d)

Here is a calculated 95% confidence interval for the mean of medv using the standard error that the bootstrap procedure yielded (0.3982063), followed by the results of a t-test procedure performed on medv.

conf.lower = mean(Boston$medv) - 2\*0.3982063  
conf.upper = mean(Boston$medv) + 2\*0.3982063  
  
conf.lower

## [1] 21.73639

conf.upper

## [1] 23.32922

t.test(Boston$medv)

##   
## One Sample t-test  
##   
## data: Boston$medv  
## t = 55.111, df = 505, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 21.72953 23.33608  
## sample estimates:  
## mean of x   
## 22.53281

The calculated confidence interval is about (21.736, 23.329), while the t-test procedure yielded a trivially wider confidence interval of (21.729, 23.336)

### Task (e)

The estimate for the population median of medv is the sample median of medv:

median(Boston$medv)

## [1] 21.2

### Task (f)

Here is a bootstrap procedure to estimate the standard error of the population median estimate (i.e., the sample mean) of medv.

median.function = function(data.vector, indices)  
 return(median(data.vector[indices]))  
  
set.seed(0)  
boot(data = Boston$medv, statistic = median.function, R=1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = median.function, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 21.2 0.0069 0.3616119

The procedure yielded an estimate of 0.3616119. This seems to be a notable degree lower than the standard error of the mean.

### Task (g)

The estimate for the population tenth percentile of medv is the sample tenth percentile of medv:

quantile(Boston$medv, 0.1)

## 10%   
## 12.75

### Task (h)

q10.function = function(data.vector, indices)  
 return(quantile(data.vector[indices], 0.1))  
  
set.seed(0)  
boot(data = Boston$medv, statistic = q10.function, R=1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = q10.function, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 12.75 0.0027 0.4990659

The procedure yielded an estimate of 0.4990659. This is markedly higher than the standard errors of the mean and median.

# Week 5 Exercises

## Chapter 6 - Exercise 9

### Task (a)

First we split the data into a training and test sample (80/20).

set.seed(0)  
training.indices = sample(dim(College)[1], dim(College)[1]\*0.8)  
  
College.train = College[training.indices,]  
College.test = College[-training.indices,]

### Task (b)

Here we fit a linear model with the training data, predicting the number of applications received based on all other variables in the data set.

We then use this model to make predictions on the test data, and report the mean squared error (MSE).

lm.model = lm(Apps ~ ., data = College.train)  
  
lm.test.pred = predict(lm.model, College.test)  
  
mean((College.test$Apps - lm.test.pred)^2)

## [1] 948328.5

### Task (c)

We will now fit a ridge regression model with a lambda chosen by cross-validation. The mean square error is reported.

train.matrix = model.matrix(Apps ~ ., College.train)[,-1]  
 test.matrix = model.matrix(Apps ~ ., College.test)[,-1]  
  
ridge.model = glmnet(train.matrix, College.train$Apps, alpha=0, lambda=10^seq(10, -2, length=100))  
  
set.seed(0)  
ridge.model.cv = cv.glmnet(train.matrix, College.train$Apps, alpha=0, lambda=10^seq(10, -2, length=100))  
  
ridge.test.pred = predict(ridge.model, s=ridge.model.cv$lambda.min, newx=test.matrix)  
mean((ridge.test.pred - College.test$Apps)^2)

## [1] 948010.3

### Task (d)

We will now fit a lasso model with a lambda chosen by cross-validation. The mean square error is reported.

train.matrix = model.matrix(Apps ~ ., College.train)[,-1]  
 test.matrix = model.matrix(Apps ~ ., College.test)[,-1]  
  
lasso.model = glmnet(train.matrix, College.train$Apps, alpha=1, lambda=10^seq(10, -2, length=100))  
  
set.seed(0)  
lasso.model.cv = cv.glmnet(train.matrix, College.train$Apps, alpha=1, lambda=10^seq(10, -2, length=100))  
  
lasso.test.pred = predict(lasso.model, s=lasso.model.cv$lambda.min, newx=test.matrix)  
mean((lasso.test.pred - College.test$Apps)^2)

## [1] 947313.9

predict(lasso.model, type="coefficients", s=lasso.model.cv$lambda.min)

## 18 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -300.45287363  
## PrivateYes -477.30306704  
## Accept 1.63565183  
## Enroll -1.02600761  
## Top10perc 54.24140904  
## Top25perc -17.50171004  
## F.Undergrad 0.06197828  
## P.Undergrad 0.05502239  
## Outstate -0.08141445  
## Room.Board 0.16002715  
## Books 0.07968076  
## Personal 0.04755280  
## PhD -9.35913985  
## Terminal -3.79319071  
## S.F.Ratio 16.33743438  
## perc.alumni 2.69361822  
## Expend 0.06202321  
## Grad.Rate 7.45242063

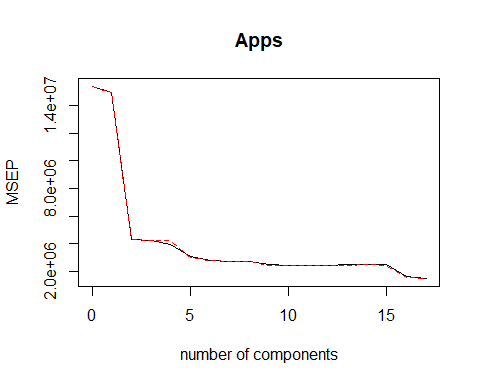
All 17 predictors have non-zero coefficients.

### Task (e)

set.seed(0)  
pcr.model = pcr(Apps ~ ., data=College.train, scale=TRUE, validation="CV")  
summary(pcr.model)

## Data: X dimension: 621 17   
## Y dimension: 621 1  
## Fit method: svdpc  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 3926 3865 2072 2060 1976 1746 1671  
## adjCV 3926 3866 2069 2056 2059 1736 1665  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1647 1641 1563 1557 1561 1561 1566  
## adjCV 1638 1634 1557 1553 1557 1558 1563  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1571 1576 1260 1223  
## adjCV 1567 1556 1250 1212  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 31.520 57.04 63.76 69.37 74.85 79.89 83.50  
## Apps 3.423 73.13 73.51 74.44 81.90 83.31 84.19  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
## X 87.04 90.26 92.76 94.92 96.77 97.88 98.76  
## Apps 84.37 85.25 85.45 85.45 85.50 85.50 85.52  
## 15 comps 16 comps 17 comps  
## X 99.37 99.83 100.00  
## Apps 90.44 92.31 92.87

validationplot(pcr.model, val.type="MSEP")



The lowest MSE produced by cross-validation is 12232 = 1495729, using all 17 principal components, but this is just regression. The more helpful minimum MSE is 15572=2424249, using M=10.

So we declare that cross-validation picks M=10, and we will compute the test error using all ncomp=10.

pcr.test.pred = predict(pcr.model, College.test, ncomp=10)  
mean((pcr.test.pred - College.test$Apps)^2)

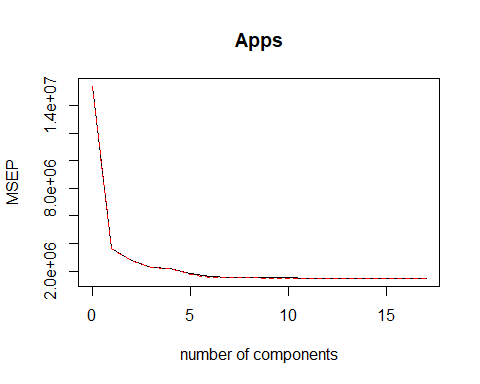
## [1] 1528997

### Task (f)

set.seed(0)  
pls.model = plsr(Apps ~ ., data=College.train, scale=TRUE, validation="CV")  
summary(pls.model)

## Data: X dimension: 621 17   
## Y dimension: 621 1  
## Fit method: kernelpls  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 3926 1907 1653 1512 1484 1348 1262  
## adjCV 3926 1903 1654 1506 1468 1331 1249  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1245 1238 1228 1225 1223 1223 1222  
## adjCV 1233 1227 1217 1214 1213 1212 1211  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1222 1223 1223 1223  
## adjCV 1212 1212 1212 1212  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 25.8 42.83 62.21 65.05 68.48 73.02 76.55  
## Apps 77.7 84.05 87.25 90.29 92.14 92.59 92.66  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
## X 79.67 81.56 84.66 88.40 90.56 92.32 93.64  
## Apps 92.73 92.81 92.84 92.85 92.86 92.87 92.87  
## 15 comps 16 comps 17 comps  
## X 96.75 97.95 100.00  
## Apps 92.87 92.87 92.87

validationplot(pls.model, val.type="MSEP")



The lowest MSE produced by cross-validation is 12222 = 1493284, using 13 principal components.

So cross-validation picks M=13, and we will compute the test error using ncomp=13.

pls.test.pred = predict(pls.model, College.test, ncomp=13)  
mean((pls.test.pred - College.test$Apps)^2)

## [1] 949978.7

### Task (g)

We seem not to be able to predict the number of college applications very well. All test errors are around 948000, with the lasso model doing the best with a test MSE of 947313.9. The PCR model did not successfully reduce dimensionality, so it’s result was much poorer, producing a test MSE of 1528997. Other than that, there isn’t that much difference.

## Chapter 6 - Exercise 11

### Task (a)

#### Create training and test samples

First we split the data into a training and test sample (80/20).

set.seed(0)  
training.indices = sample(dim(Boston)[1], dim(Boston)[1]\*0.8)  
  
Boston.train = Boston[training.indices,]  
Boston.test = Boston[-training.indices,]

#### Linear Model

Here we fit a linear model with the training data, predicting towns’ per capita crime rate based on all other variables in the data set.

We then use this model to make predictions on the test data, and report the mean squared error (MSE).

lm.model = lm(crim ~ ., data = Boston.train)  
  
lm.test.pred = predict(lm.model, Boston.test)  
  
mean((Boston.test$crim - lm.test.pred)^2)

## [1] 40.89623

#### Ridge Regression Model

We will now fit a ridge regression model with a lambda chosen by cross-validation. The mean square error is reported.

train.matrix = model.matrix(crim ~ ., Boston.train)[,-1]  
 test.matrix = model.matrix(crim ~ ., Boston.test)[,-1]  
  
ridge.model = glmnet(train.matrix, Boston.train$crim, alpha=0, lambda=10^seq(10, -2, length=100))  
  
set.seed(0)  
ridge.model.cv = cv.glmnet(train.matrix, Boston.train$crim, alpha=0, lambda=10^seq(10, -2, length=100))  
  
ridge.test.pred = predict(ridge.model, s=ridge.model.cv$lambda.min, newx=test.matrix)  
mean((ridge.test.pred - Boston.test$crim)^2)

## [1] 41.07925

#### Lasso method

We will now fit a lasso model with a lambda chosen by cross-validation. The mean square error is reported.

train.matrix = model.matrix(crim ~ ., Boston.train)[,-1]  
 test.matrix = model.matrix(crim ~ ., Boston.test)[,-1]  
  
lasso.model = glmnet(train.matrix, Boston.train$crim, alpha=1, lambda=10^seq(10, -2, length=100))  
  
set.seed(0)  
lasso.model.cv = cv.glmnet(train.matrix, Boston.train$crim, alpha=1, lambda=10^seq(10, -2, length=100))  
  
lasso.test.pred = predict(lasso.model, s=lasso.model.cv$lambda.min, newx=test.matrix)  
mean((lasso.test.pred - Boston.test$crim)^2)

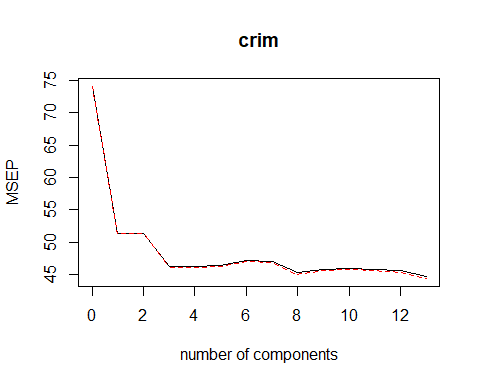
## [1] 40.91244

#### PCR method

set.seed(0)  
pcr.model = pcr(crim ~ ., data=Boston.train, scale=TRUE, validation="CV")  
summary(pcr.model)

## Data: X dimension: 404 13   
## Y dimension: 404 1  
## Fit method: svdpc  
## Number of components considered: 13  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 8.609 7.170 7.164 6.797 6.797 6.809 6.867  
## adjCV 8.609 7.167 7.161 6.790 6.790 6.803 6.857  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 6.860 6.728 6.768 6.779 6.771 6.753 6.680  
## adjCV 6.849 6.712 6.754 6.764 6.755 6.736 6.663  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 47.48 60.28 69.36 76.43 83.17 88.22 91.45  
## crim 31.22 31.54 39.09 39.35 39.43 39.70 40.03  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## X 93.66 95.60 97.28 98.59 99.55 100.00  
## crim 42.58 42.58 43.20 43.45 44.24 45.41

validationplot(pcr.model, val.type="MSEP")



The lowest MSE produced by cross-validation is 6.682 = 44.6224 uses all 13 principal components, but that is just regression. The next lowest MSE, however, uses M=8 and is 6.7282 = 45.266.

So we declare that cross-validation picks M=8, and we will compute the test error using all ncomp=8.

pcr.test.pred = predict(pcr.model, Boston.test, ncomp=8)  
mean((pcr.test.pred - Boston.test$crim)^2)

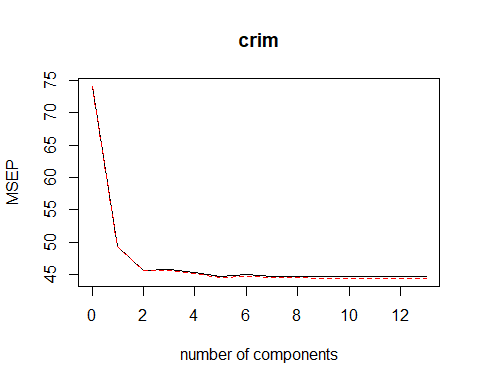
## [1] 42.70996

#### PLSR method

set.seed(0)  
pls.model = plsr(crim ~ ., data=Boston.train, scale=TRUE, validation="CV")  
summary(pls.model)

## Data: X dimension: 404 13   
## Y dimension: 404 1  
## Fit method: kernelpls  
## Number of components considered: 13  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 8.609 7.022 6.757 6.770 6.736 6.690 6.704  
## adjCV 8.609 7.019 6.748 6.751 6.717 6.674 6.686  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 6.687 6.686 6.682 6.680 6.680 6.680 6.680  
## adjCV 6.669 6.668 6.664 6.662 6.663 6.663 6.663  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 47.09 56.43 62.26 71.23 76.16 79.68 83.84  
## crim 34.58 41.77 43.86 44.60 44.94 45.22 45.33  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## X 86.73 89.9 95.02 96.70 98.06 100.00  
## crim 45.38 45.4 45.40 45.41 45.41 45.41

validationplot(pls.model, val.type="MSEP")



The lowest MSE produced by cross-validation is 6.682 = 44.6224, using 10 or more principal components.

So cross-validation picks M=10, and we will compute the test error using ncomp=10.

pls.test.pred = predict(pls.model, Boston.test, ncomp=10)  
mean((pls.test.pred - Boston.test$crim)^2)

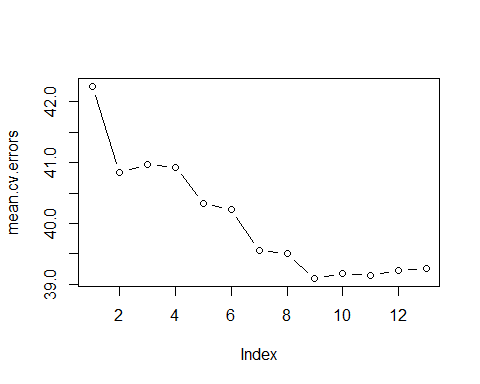
## [1] 40.87073

We will now use a best subset selection procedure, allowing the algorithm to consider models with up to 13 predictors.

# The regsubsets predict function, courtesy of ISLR  
predict.regsubsets =function(object, newdata, id,...){  
 form = as.formula(object$call[[2]])  
 mat = model.matrix(form, newdata)  
 coefi = coef(object, id=id)  
 xvars = names(coefi)  
 mat[,xvars]%\*%coefi  
}  
  
k=10  
set.seed(0)  
folds = sample(1:k, nrow(Boston), replace=TRUE)  
cv.errors = matrix(NA, k, 13, dimnames=list(NULL, paste(1:13)))  
  
for(j in 1:k){  
 best.fit = regsubsets(crim ~ ., data=Boston[folds!=j,], nvmax=13)  
 for(i in 1:13){  
 pred=predict(best.fit, Boston[folds==j,], id=i)  
 cv.errors[j,i]= mean((Boston$crim[folds==j] - pred)^2)  
 }  
}  
  
mean.cv.errors=apply(cv.errors, 2, mean)  
mean.cv.errors

## 1 2 3 4 5 6 7 8   
## 42.25616 40.83932 40.96665 40.92546 40.33157 40.23861 39.55875 39.50477   
## 9 10 11 12 13   
## 39.09773 39.17212 39.14737 39.22137 39.26521

par(mfrow=c(1,1))  
plot(mean.cv.errors, type='b')



Cross-validation selects a 9-variable model, which yields a mean cross-validation error of 39.09773.

### Task (b)

Here is a summary of the test MSEs produced by the different methods we used:

40.89623 Linear model 41.07925 Ridge regression 40.91244 Lasso method 42.70996 PCR method 40.87073 PLS method 39.09773 Best subsets method

All methods produced MSEs around 40. The PCR method is marginally worse, and the Best subsets method is marginally the best. My final choice would be the model produced by the best subsets method because it has the best MSE and is the most interpretable.

### Task (c)

My choice only uses 9 of the 13 available predictors because the other 4 proved not to be helpful after cross-validation. In other words, when models containing these extraneous predictors were tested, they proved not to be informative for the crim variable–predictions that used more than 9 variables actually had higher test MSE.