1 StatLearn Practical exercise 5

Exercise 1.1. *Download the LA ozone data set from the book homepage. We will be regressing the cube root of the ozone concentration on the other variables. Divide the data set into two groups at random. One group, which we call the training data, containing 2/3 of the observations and one group, which we call the test data, with 1/3 of the observations. Use only the training data below for the estimation.*

Solution. We first load the necessary libraries for ridge regression, LASSO and best subset selection.

```
library(MASS)
library(glmnet)
library(leaps)
```

After this, we load the data set. The data set contains 330 observations, each consisting of 9 predictors and 1 response. The response, `ozone`, is in the first column. As we wish to regress the cubic root of `ozone`, we update this column with the cubic roots of the original data.

```
LAozone = read.table("http://www-stat.stanford.edu/
                      ~tibs/ElemStatLearn/datasets/LAozone.data",
                      sep=",",head=T)
LAozone[,1]<-LAozone[,1]^(1/3)
```

We divide the data sets into training and test parts. We use a training sample of 40, as this yields results which better demonstrate the points of the following exercises.

```
trainNumObs<-40
trainIndex<-sample(1:330,size=trainNumObs,replace=FALSE)
LAozoneTrain<-LAozone[trainIndex,]
LAozoneTest<-LAozone[-trainIndex,]
```

□

Exercise 1.2. *Compute the best model for each dimension (best subset selection), estimate test error and compute the training error. Make a plot.*

Solution. We first define the design matrices and response vectors for the training and test data sets for later use.

```
trainDesign<-as.matrix(LAozoneTrain[,-1])
trainResponse<-as.matrix(LAozoneTrain[,1])
testDesign<-as.matrix(LAozoneTest[,-1])
testResponse<-as.matrix(LAozoneTest[,1])
```

In order to compute the best subset linear regression estimates, we use the function `regsubsets` from the `leaps` package. `nbest=1` signifies that we are only interested in the one best subset for each number of predictors, instead of, say, the 10 best subsets. `nvmax=9` signifies that we wish to have best subsets estimates up to 9 predictors.

```
LABestSubset<-regsubsets(x=trainDesign,y=trainResponse,nbest=1,nvmax=9)
bestSubsetCoef<-coefficients(LABestSubset,1:9)
```

We need to compute the training and the test errors. These are given as the empirical mean loss of the estimated predictor function over the training and the test samples for each model, that is, for each best subset up to size 9. The `i`'th best subset estimates are stored as a vector in `bestSubsetCoef[[i]]`. The loss used in linear regression is the squared loss, and therefore, the training error is the mean square of the residuals.

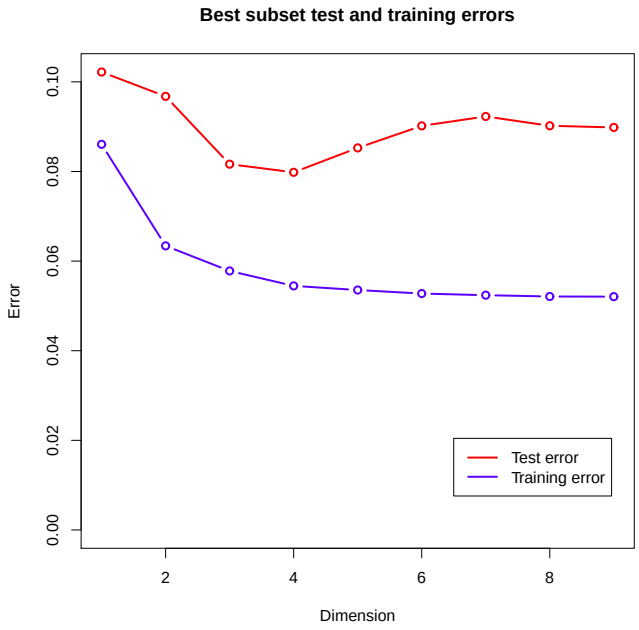
```
bestSubsetTestError<-numeric()
bestSubsetTrainingError<-numeric()
for(i in 1:9)
{
param<-rep(0,9)
names(param)<-c("vh","wind","humidity","temp","ibh","dpg","ibt","vis","doy")
param[names(bestSubsetCoef[[i]][-1])]<-bestSubsetCoef[[i]][-1]

bestSubsetTestError[i]<-mean((testResponse-
                                    bestSubsetCoef[[i]][1]-
                                    testDesign%*%param)^2)
bestSubsetTrainingError[i]<-mean((trainResponse-
                                        bestSubsetCoef[[i]][1]-
                                        trainDesign%*%param)^2)
}
```

Finally, we plot the test error and the training errors.

```
plot(1:9,ylim=c(0,max(bestSubsetTestError,bestSubsetTrainingError)),type="n",
     xlab="Dimension",ylab="Error",main="Best subset test and training errors")
points(1:9,bestSubsetTestError,type="b",col="red",lwd=2)
points(1:9,bestSubsetTrainingError,type="b",col="blue",lwd=2)
```

```
legend(x=6.5,y=max(bestSubsetTestError,bestSubsetTrainingError)/5,
       c("Test error","Training error"),col=c("red","blue"),lwd=2)
```

The results of the plot are as follows.



We note that the training error is decreasing as a function of the dimension of the fit. This corresponds to better fitting to our training data as a higher number of predictors are made available. The test error is not decreasing, this corresponds to our fit for the training data set not necessarily generalizing well to the test data set. Furthermore, the training error is systematically lower than the test error. This corresponds to the "optimism of the training error", meaning that our fit does better on our particular training set than it does on other datasets even with similar distributions. □

Exercise 1.3. *Compute the ridge and LASSO paths. Estimate the test error, compute the training error and plot them as a function of the penalization parameter.*

Solution. In order to calculat the LASSO path, we use the function `glmnet` from the package `glmnet`. `glmnet` optimizes $\frac{1}{2n}\|y - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1$, where n is the number of observations, and y and $\mathbf{X}$ are the standardized response and design matrix. Note that this does not generally correspond to any particular L^1-penalized optimization problem in the nonstandardized case.

`glmnet` returns an object `LAlasso` of class `glmnet`, where the coefficients may be obtained as `coef(LAlasso)`, and the sequence of λ's may be obtained as `LAlasso$lambda`.

```
LAlasso<-glmnet(trainDesign,trainResponse,
                family="gaussian",alpha=1,standardize=TRUE)
lassoCoef<-t(as.matrix(coef(LAlasso)))
lassoLambdaSeq<-LAlasso\$lambda
```

In order to obtain the ridge regression solutions, we use the function `lm.ridge` from the package `MASS`. `lm.ridge` does not automatically generate the sequence of λ's to be used. In order to obtain a reasonable sequence of λ's, experience shows that the sequence generated by `glmnet` times `4*trainNumObs` yields reasonable results. It is essential to use `coef(LAridge)` and not `LAridge$coef`, as the latter returns estimates from a ridge regression problem which is on a different scale than our original ridge regression problem.

```
ridgeLambdaSeq<-4*trainNumObs*lassoLambdaSeq
LAridge<-lm.ridge(ozone ~ .,data=LAozoneTrain,lambda=ridgeLambdaSeq)
ridgeCoef<-as.matrix(coef(LAridge))
```

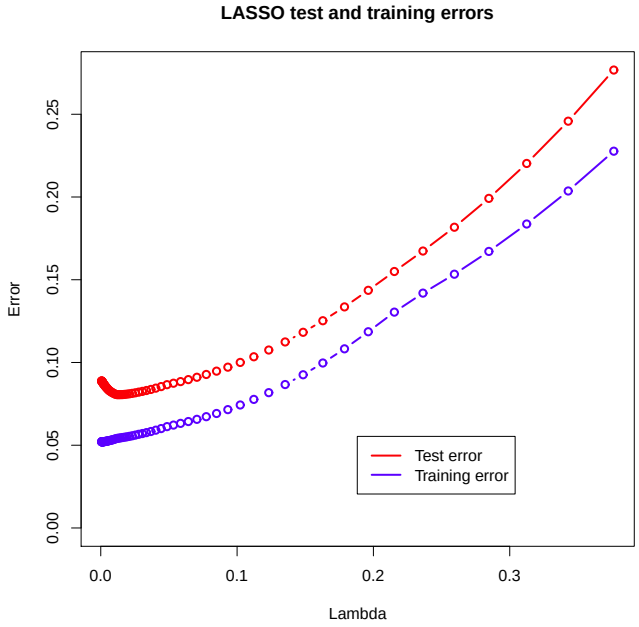
Test and training errors are then calculated by manually calculating the fitted values, as in the best subset case, and squaring and averaging the resulting residuals.

```
lassoTestError<-numeric()
lassoTrainingError<-numeric()
ridgeTestError<-numeric()
ridgeTrainingError<-numeric()
for(i in 1:length(lassoLambdaSeq))
{
lassoTestError[i]<-mean((testResponse-lassoCoef[i,1]-
                                testDesign%*%lassoCoef[i,-1])^2)
lassoTrainingError[i]<-mean((trainResponse-lassoCoef[i,1]-
                                    trainDesign%*%lassoCoef[i,-1])^2)
}

for(i in 1:length(ridgeLambdaSeq))
{
ridgeTestError[i]<-mean((testResponse-ridgeCoef[i,1]-
                                testDesign%*%ridgeCoef[i,-1])^2)
ridgeTrainingError[i]<-mean((trainResponse-ridgeCoef[i,1]-
                                    trainDesign%*%ridgeCoef[i,-1])^2)
}
```

Next, we plot the test and training errors against the sequence of λ's for the LASSO fit.

```
X11()
plot(lassoLambdaSeq,lassoLambdaSeq,
     ylim=c(0,max(lassoTestError,lassoTrainingError)),type="n",
     xlab="Lambda",ylab="Error",main="LASSO test and training errors")
points(lassoLambdaSeq,lassoTestError,type="b",col="red",lwd=2)
points(lassoLambdaSeq,lassoTrainingError,type="b",col="blue",lwd=2)
legend(x=max(lassoLambdaSeq)/2,y=max(lassoTestError,lassoTrainingError)/5,
       c("Test error","Training error"),col=c("red","blue"),lwd=2)
```

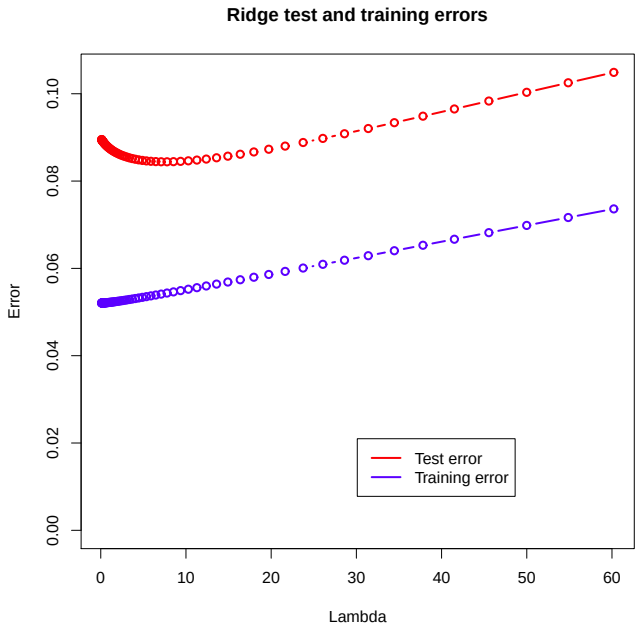
The results of the plot are as follows.



Compared to the best subset case, the training error now appears as an increasing function. This is because the effective dimensionality of the model decreases with values plotted on the horizontal axis, λ, while it in the best subset case increased as the values plotted on the horizontal axis, subset size, increased. The test error increases for very small and very large values of λ. The increase for very small values of λ appears because the models for small values of λ overfit and thus do not generalize well. The increase for very large values of λ appears because the models have small dimensionality and therefore have difficulty fitting the data well. We repeat the exercise for the ridge regression case.

```
X11()
plot(ridgeLambdaSeq,ridgeLambdaSeq,
     ylim=c(0,max(ridgeTestError,ridgeTrainingError)),type="n",
     xlab="Lambda",ylab="Error",main="Ridge test and training errors")
points(ridgeLambdaSeq,ridgeTestError,type="b",col="red",lwd=2)
points(ridgeLambdaSeq,ridgeTrainingError,type="b",col="blue",lwd=2)
legend(x=max(ridgeLambdaSeq)/2,y=max(ridgeTestError,ridgeTrainingError)/5,
       c("Test error","Training error"),col=c("red","blue"),lwd=2)
```

This results in the following plot.



As in the LASSO case, the training error increases as the penalization parameter λ inceases, corresponding to a greater training error for models with less dimensionality. The test error increases for very small and very large values of λ, corresponding to overfitting for small λ and difficultly with fitting the data for large λ. □

Exercise 1.4. *Compute the C_p statistic for best subset selection and ridge regression and compare with the estimated test errors.*

Solution. The C_p statistic is generally given as

$$C_p = \overline{\text{err}}_\text{s} + \frac{2d}{n}\hat{\sigma}_\varepsilon^2,$$

where $\overline{\text{err}}_\text{s}$ is the training error, $\hat{\sigma}_\varepsilon^2$ is a suitable noise variance estimate, colloquially denoting an estimate of the conditional distribution of Y given X, n is the number of training observations and d is the effective degrees of freedom. We have already calculated the training errors for the best subset and ridge regression models. It remains to calculate the effective degrees of freedom and the noise variance estimate.

We may obtain the noise variance estimate by unbiased estimation of the variance in the ordinary linear regression model. This is the sum of squared residuals divided by the number of observations, `trainNumObs`, minus the dimension of the parameter space, 9.

```
noiseVar<-1/(trainNumObs-9)*sum(lm(ozone ~ .,data=LAozoneTrain)\$residuals^2)
```

In the best subset case, the effective degrees of freedom is equal to the subset size, while in the ridge regression case, the effective degrees of freedom is $\text{tr}\, \mathbf{X}(\mathbf{X}^t\mathbf{X} + \lambda I)^{-1}\mathbf{X}^t$, where $\mathbf{X}$ is the design matrix of the ridge regression problem, λ is the penalization parameber and I is the identity matrix of order corresponding to the number of predictors.

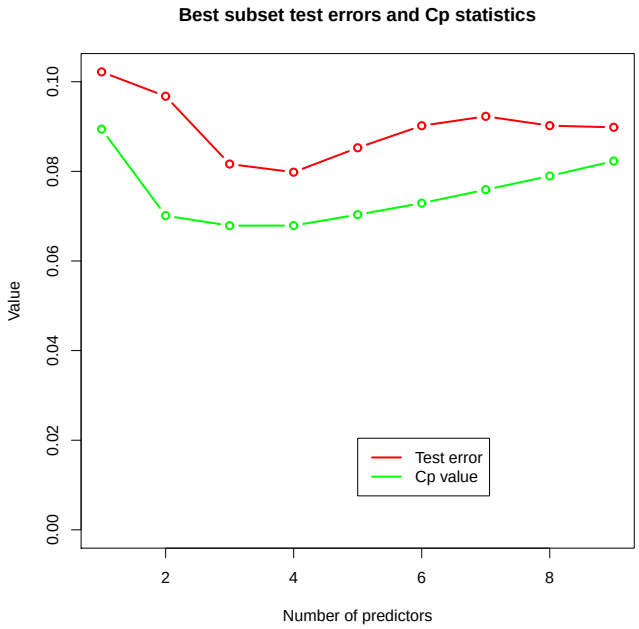
We calculate the C_p values. The ridge regression estimates obtained through `lm.ridge` are optimizers of $\|y - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_2^2$, where $\mathbf{X}$ is the standardized design matrix. Therefore, when calculating the ridge regression effective degrees of freedom, we need to take this into account.

```
trainDesignStandard<-scale(trainDesign)
bestSubsetCp<-numeric()

for(i in 1:9) bestSubsetCp[i]<-bestSubsetTrainingError[i] +
                               2 * i / trainNumObs * noiseVar
for(i in 1:length(ridgeLambdaSeq))
{
df<-sum(diag(trainDesignStandard%*%
                solve(t(trainDesignStandard)%*%
                        trainDesignStandard+
                        ridgeLambdaSeq[i]*diag(1,9))%*%
                t(trainDesignStandard)))
ridgeCp[i]<-ridgeTrainingError[i] + 2 * df / trainNumObs * noiseVar
}
```

Next, we plot the C_p values for best subset regression as well as the test errors obtained earlier.

```
X11()
plot(1:9,1:9,ylim=c(0,max(bestSubsetTestError,bestSubsetCp)),type="n",
     xlab="Number of predictors",ylab="Value",
     main="Best subset test errors and Cp statistics")
points(1:9,bestSubsetTestError,type="b",col="red",lwd=2)
points(1:9,bestSubsetCp,type="b",col="green",lwd=2)
legend(x=5,y=max(bestSubsetTestError,bestSubsetCp)/5,
       c("Test error","Cp value"),col=c("red","green"),lwd=2)
```

This results in the following plot for best subset regression.



Ideally, C_p estimates the in-sample error, which is defined for any fixed set of independent variable observations (x_i) as

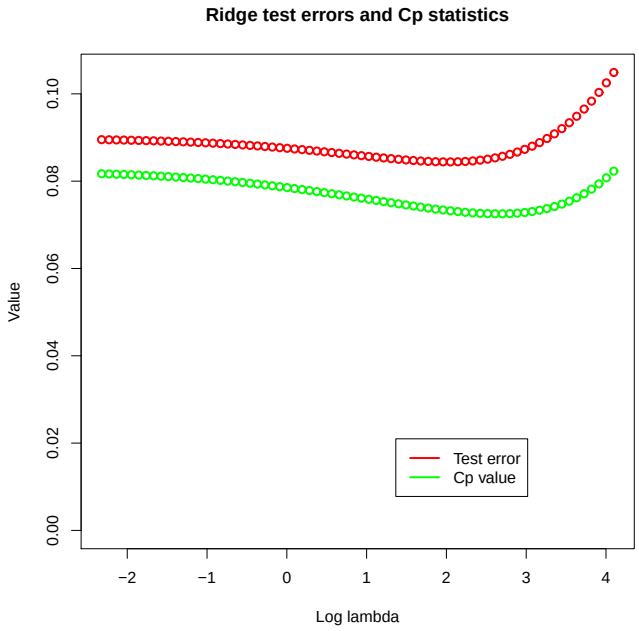
$$\mathrm{Err}_{\mathrm{in}} = \frac{1}{n}\sum_{i=1}^{n} EL(Y_i', \hat{f}_s(x_i)),$$

where $\hat{f}_s(x_i)$ is the fitted value at x_i based on stochastic responses (Y_i), and Y_i' has the distribution of Y given $X = x_i$, and (Y_i) and (Y_i') are independent. The test error does not necessarily estimate the in-sample error unbiasedly, as the test error is based on the estimate using the training data, while exchanging both the response and the independent variables

with new data. Still, we would hope that the test errors are reminiscent of the in-sample error, and in particular that the C_p values and the test errors approximately agree. This is somewhat validated by the plot results, although the results are highly dependent on the training sample. Next, we plot the ridge regression C_p values and test errors.

```
X11()
plot(log(ridgeLambdaSeq),ridgeLambdaSeq,ylim=c(0,max(ridgeTestError,ridgeCp)),
     type="n",xlab="Log lambda",ylab="Value",
     main="Ridge test errors and Cp statistics")
points(log(ridgeLambdaSeq),ridgeTestError,type="b",col="red",lwd=2)
points(log(ridgeLambdaSeq),ridgeCp,type="b",col="green",lwd=2)
legend(x=max(log(ridgeLambdaSeq))/3,y=max(ridgeTestError,ridgeCp)/5,
       c("Test error","Cp value"),col=c("red","green"),lwd=2)
```

The plot obtained from this is the following. As before, the C_p values and the test errors are



relatively close. In both cases, we note that the C_p values are systematically lower than the test errors. This may be due to small-sample issues or to a discrepancy between the mean of the test error and the in-sample error. □