# Compulsory exercise 2: Group XX

TMA4268 Statistical Learning V2018

*NN1, NN2 and NN3*

*12 mars, 2018*

## 1a)

- Q1: We have $d$ possible predictors. First of all, we can chose how many of these predictors we want to use. We have the choice to use $k = 0, 1, 2, \ldots, d$ predictors in our model. If we use $k = 0$ predictors, we don't really have a linear regression model, instead we will get a model that predicts new samples as the average of all previous samples. Even though this is not a linear regression model, we want to investigate whether this gives a better result than using a linear regression model would. That would imply that there is no linear connection between the data from our sample. Let us now focus on the choices of $k$ that results in linear regression models. If we choose to use $k = 1$ predictor, we again have a choice in which of the $d$ possible predictors we want to use. Obviuosly there are $d$ possible choices for $k = 1$, but if we want to be more systematic we can say that we choose $k$ different predictors from $d$ possible ones, and we can not choose the same twice. Hence, the number of possible combinations using $k$ predictors is given by the standard nCr-formula $\binom{d}{k}$. Now we have our $d$ different choices of $k$, which all gives different linear regression models. In total, the number of models is

$$\binom{d}{1} + \binom{d}{2} + \cdots + \binom{d}{d}.$$

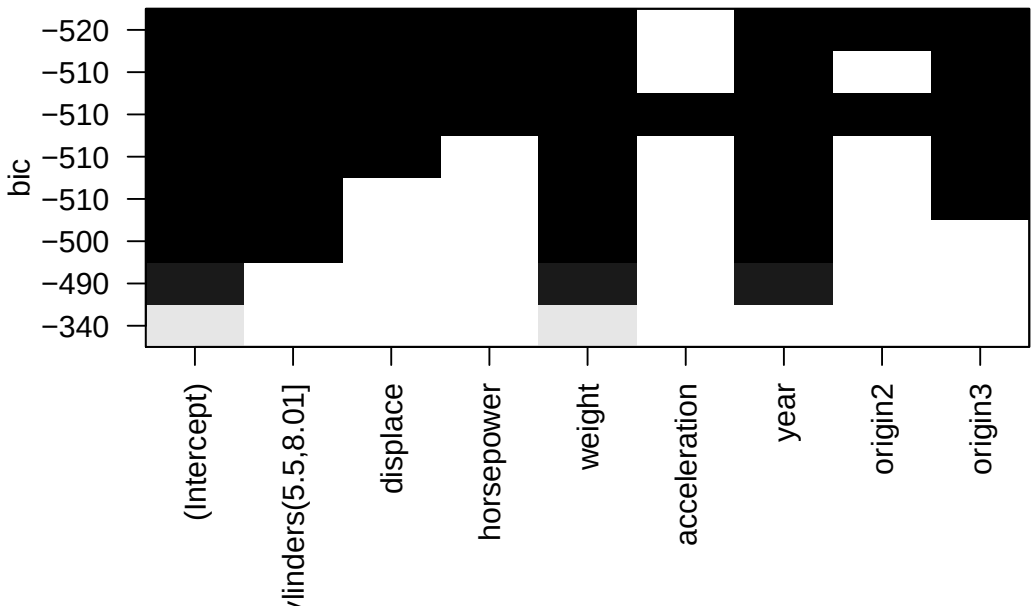
. The binomial theorem states that

$$(1 + x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k.$$

. By setting $x = 1$, we see that the number of different linear regression models is given by

$$\sum_{k=1}^{d} \binom{d}{k} = 2^d - 1.$$

- Q2: The following algorithm illustrates how the best subset method finds the best model amongst the $2^d - 1$ possible ones: Denote first by $\mathcal{M}_0$ the model using no predictors. This model simply predicts the sample mean for new observations. Next, for $k = 1, 2, \ldots, d$, fit all $\binom{d}{k}$ models, by solving the equation $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ only contains the predictors for one specific model. Calculate the RSS, and choose the model with the lowest RSS amongst the $\binom{d}{k}$ models, to be considered later. Now after the for-loop, we have $d+1$ possible models to consider. Calculate $BIC = \frac{1}{n}(RSS + log(d)k\hat{\sigma}^2)$ for each, and choose the model with the lowest $BIC$. This is the best model by the best subset method, using $BIC$-criterion. $R^2$ is given by $1 - RSS/TSS$. In this expression, we do not care how many predictors are being used. This is a problem, because training error is a poor estimate for test error, since it may cause overfitting. A possible solution to this is to add penalties, depending on how many predictors we use. If we were to use $R^2$ instead of $BIC$ to choose the best model, our best model would surely contain all $d$ predictors since over-fitting would make our model fit the training data better. This is unwanted as we would rather have a model that best fit test data, which $BIC$ better simulates.




**1b)**

- Q3: As explained in Q2, $RSS$ is calculated for all models, and for each model complexity, we choose the model with the lowest $RSS$ as our best model. The `regsubsets`-function in R performs the actual calculations, and in the summary we can see which predictors corresponds to the lowest $RSS$ for each model complexity. We see from this summary, which is printed in the task, that the best model with two predictors use the "weight" and "year" columns.
- Q4: Now assuming we have the best model for each model complexity, we still want to choose which model complexity to use. As discussed in Q2, we would have the smallest $RSS$ for the highest possible model complexity in the training set, but since we want to avoid overfitting, we add penalties to more complex models. Therefore, for each model complexity, we add the penalties according to the $BIC$-criterion, and then choose the model with the lowest value as our best method. The *regsubsets*-function in R also does the calculations for the penalties, using $BIC$-criterion in addition to other penalty-adding methods. Thus we can from the `regsubsets`-function also retrieve the $BIC$-values for each model complexity, and choose the lowest one as our model. The figure above, which is the result of plotting the `regsubsets`-data also contains the necessary information for choosing the best model. On the $y$ scale, we see $BIC$-values for the models of different complexity, and on the $x$-scale we see which predictors are contained in the best model of this complexity, as a black box. We choose the model with the lowest $BIC$-value, which we see has $BIC \sim -520$, and contains seven predictors, all but acceleration. Hence, this is our best model according to the best subset method using $BIC$-criterion on the training data. From the prints in the task we see th more exact $BIC = -516.9417$, which was rounded off to -520. The following R-code fits this model to the training data, by using the `lm`-function. The summary output gives us information about how well the model fit. We see that all of the $t$-values are fairly high, and the $p$-values for all but one are within the 0.001-significance level, while the last one is within the 0.01 significance level. We can therefore with a very high certainty throw away the zero hypothesis without making a mistake. We also see R-squared-values relatively close to 1, which indicates a good

model fit. Also we have $RSS = RSE^2(n-k-1)$, so that

$$MSE = \frac{RSS}{n} = \frac{RSE^2(n-k-1)}{n} = \frac{3.233^2 \cdot 305}{313} = 10.185.$$

We have also performed an Anderson-Darling test for Normality. The $p$-value for this test is $9.241 \cdot 10^-6$, which is very small, and indicates that our data does not have normal distributed errors. ???

```
library(nortest)
trainLm=lm(mpg~. -acceleration, data=ourAutoTrain)
summary(trainLm)
ad.test(rstudent(trainLm))
yhat=predict(trainLm,ourAutoTest)
mean((ourAutoTest$mpg-yhat)^2)
```

```
##
## Call:
## lm(formula = mpg ~ . - acceleration, data = ourAutoTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7254 -2.0561 -0.3412  1.7122 12.9550
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.941e+01  4.589e+00  -4.230 3.09e-05 ***
## cylinders(5.5,8.01] -3.533e+00  7.469e-01  -4.730 3.43e-06 ***
## displace             3.279e-02  6.749e-03   4.858 1.90e-06 ***
## horsepower          -4.883e-02  1.184e-02  -4.124 4.80e-05 ***
## weight              -5.824e-03  6.185e-04  -9.415  < 2e-16 ***
## year                 7.849e-01  5.699e-02  13.771  < 2e-16 ***
## origin2              1.794e+00  6.204e-01   2.892  0.00411 **
## origin3              2.906e+00  5.943e-01   4.891 1.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.233 on 305 degrees of freedom
## Multiple R-squared:  0.8344, Adjusted R-squared:  0.8306
## F-statistic: 219.6 on 7 and 305 DF,  p-value: < 2.2e-16
##
##
##  Anderson-Darling normality test
##
## data:  rstudent(trainLm)
## A = 2.2684, p-value = 9.241e-06
##
## [1] 8.931018
```

- Q5: The code above also predicts new values for the test set, and calculates the test-$MSE$. Here, we got $MSE = 8.931$, which actually is smaller than the training $MSE$, indicating that we indeed have managed to avoid overfitting, and made a model which predicts new values pretty well.

## 1c)

- Q6. K-fold cross-validation involves dividing the data into $k$ folds, $C_1, C_2, \ldots, C_k$. We then want to divide the data into a training and test set, and we choose one of the $k$ parts as the test set, while we

let the rest be training data. After fitting the model, we calculate $MSE$ for the specific test folder, as

$$MSE_j = \frac{a}{n_k} \sum_{i \in C_j} (y_i - \hat{y}_i)^2$$

- Q7. Why may $k$-fold cross-validation be preferred to leave-one-out cross-validation?

## 1d

- Q8. The R-code for performing 10-fold cross validation is below.

```
library(caret)
library(leaps)
predict.regsubsets=function(object,newdata,id){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
k=10
set.seed(4268)
folds=sample(1:k,nrow(ourAutoTrain),replace=TRUE)
cv.errors=matrix(NA,k,8,dimnames=list(NULL,paste(1:8)))
for (j in 1:k){
  best.fit=regsubsets(mpg~.,data=ourAutoTrain[folds!=j,])
  for (i in 1:8){
    pred=predict.regsubsets(best.fit,ourAutoTrain[folds==j,],id=i)
    cv.errors[j,i]=mean((ourAutoTrain$mpg[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
which.min(mean.cv.errors)
```

```
##        1        2        3        4        5        6        7        8
## 20.04605 11.98844 11.92601 11.29254 11.71515 10.72818 10.56209 10.61032
## 7
## 7
```

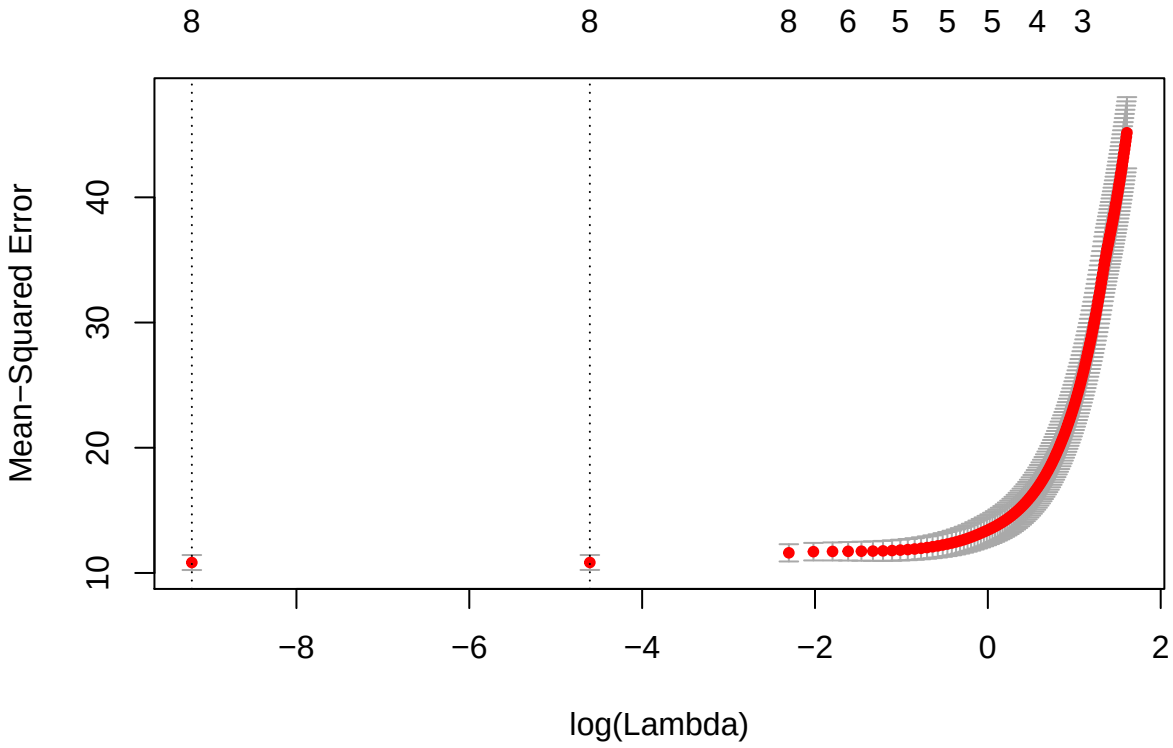
- Q9. We see from the output of the code the $MSE$ for the different complexities, and we also see we get the lowest $MSE$ when using seven predictors. Hence, we want to use the same model complexity as we did for the $BIC$-criterion.

```
# MSE on test set
```

- Q10. Now that we established which model complexity we want to use, we would like to fit this on our entire training set, to get the best model possible. We would therefore use the `regsubsets`-function on the training set, and choose the model using seven predictors. We do however note that this will give us the exact same model as we did for the $BIC$-criterion. Our comments on the model fit for the training set, and the $MSE$ for the training set, is therefore identical to our results in Q4. Also, our predicted new values for the test set, and the MSE for the test set is identival to our results in Q5. We therefore refer to Q4 and Q5 in this task.

## 2a) Explain figures

- Q11: Which figure (1 or 2) corresponds to ridge and which figure corresponds to lasso?
- Q12. Use the two figures and the above formulas to explain...
- Q13. Can you use lasso and/or ridge regression to perform model selection similar to what you did in Problem 1?




## 2b) Finding the optimal $\lambda$

- Q14: Explain what the function `cv.glmnet` does.
- Q15. Explain what we see in the above plot.
- Q16: Finding the optimal lambda:

```
# need some R code here
```

## 3c) Prediction

- Q17: Fit model, coefficients,...

```
# fit the lasso
```

```
# 0 for cylinder, displace, horsepower, weight, acceleration, year, 0 for origin2 and 0 for origin3
newx=matrix(c(0,150,100,3000,10,82,0,0),nrow=1)
# then do the prediction
```

- Q18: Predicted value:

### 3a)

- Q19: Fitting the specified `gam`

```
library(gam)
# write R code
```

- Q20: The cubic spline basis.