Compulsory exercise 2: Group 18

TMA4268 Statistical Learning V2022

Thomas Rødland, Erlend Lokna

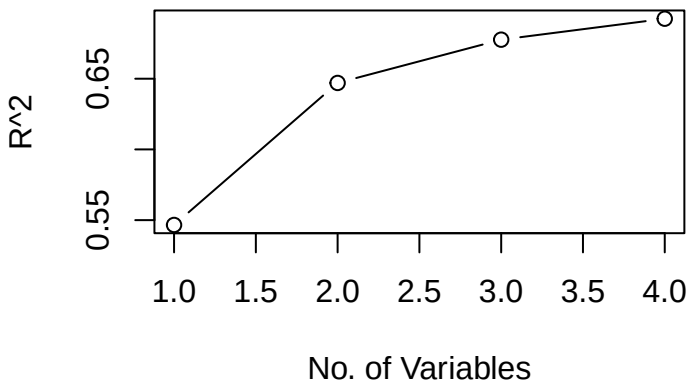
25 mars, 2022

```
set.seed(1)
boston <- scale(Boston, center=T, scale=T)

# split into training and rest sets
train.ind = sample(1:nrow(boston), 0.8 * nrow(boston))
boston.train = data.frame(boston[train.ind, ])
boston.test = data.frame(boston[-train.ind, ])
```

Problem 1

a)

```
#forward subset selection:
regfit_fwd.full = regsubsets(medv~., data = boston.train, nvmax=13, method = "forward")
regfit_fwd.four = regsubsets(medv~., data = boston.train, nvmax=4, method = "forward")

#plotting the R-squared vs number of predictors
plot(summary(regfit_fwd.four)$rsq, xlab='No. of Variables',ylab='R^2',type='b')
```



b)

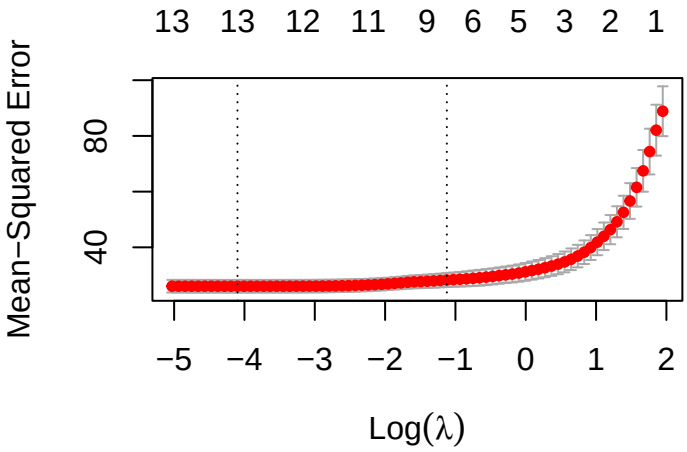
```
#the best four predictors:
coef(regfit_fwd.four, 4)
```

```
## (Intercept)          rm         dis     ptratio       lstat
##  0.02268276  0.34728504 -0.14653616 -0.21570329 -0.53034047
```

c)

```
#K-fold cross-validation (k=5)
#setup:
df.X.std <- scale(dplyr::select(Boston, -medv))
X.train <- as.matrix(df.X.std)[train.ind,] #converting to matrix.
X.test <- as.matrix(df.X.std)[-train.ind,]
Y.train <- Boston[train.ind, "medv"]
Y.train = as.matrix(Y.train)
Y.test <- Boston[-train.ind, "medv"]
Y.test = as.matrix(Y.test)

#calculation using cv.glmnet():
kf5_cv = cv.glmnet(x=X.train, y=Y.train, alpha = 1, nfolds = 5)
```

```
#plot:
plot(kf5_cv)
```



Lambda corresponding to minimal Mean-Squared error:

```
kf5_cv$lambda.min
```

```
## [1] 0.01658476
```

coefficients:

```
coef(kf5_cv, s=kf5_cv$lambda.min)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
```

```
## (Intercept) 22.74991687
## crim        -0.75916622
## zn           0.87882705
## indus        0.06224523
## chas         0.80268368
## nox         -1.62765265
## rm           2.87560832
## age         -0.10735056
## dis         -2.92307108
## rad          2.52855436
## tax         -1.95602522
## ptratio     -1.88338836
## black        0.95034028
## lstat       -3.94210652
```

d)

1. True
2. False
3.
4. True

Problem 2

```
set.seed(1)

# load a synthetic dataset
id <- "1CWZYfrLOrFdrIZ6Hv73e3xxt0SFgU4Ph"  # google file ID
synthetic <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id))

# split into training and test sets
train.ind = sample(1:nrow(synthetic), 0.8 * nrow(synthetic))
synthetic.train = data.frame(synthetic[train.ind, ])
synthetic.test = data.frame(synthetic[-train.ind, ])

# show head(..)  Y: response variable; X: predictor variable
head(synthetic)
```

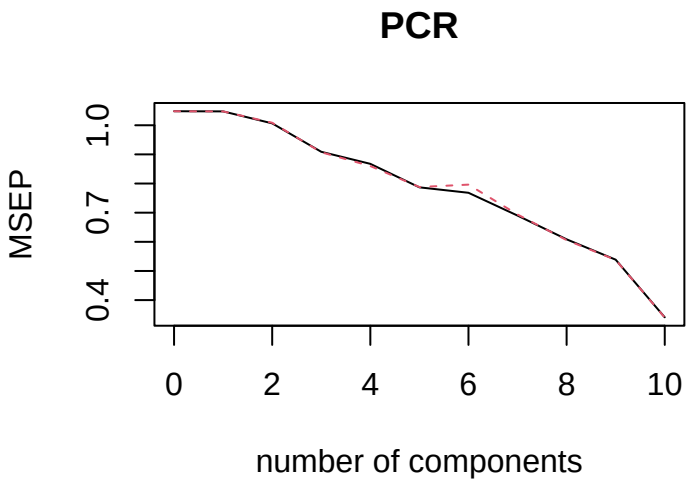
```
##             Y          X1          X2         X3         X4         X5
## 1 -1.43753239 -0.75905055 -0.69720326 -0.3016852 -0.7434697  0.8807558
## 2 -1.70972989 -0.28635632  0.04809182  0.5791725 -0.7446170  0.9935311
## 3  1.33931240  0.09574117 -0.89605758 -0.9636347  0.5554647 -0.5341800
## 4  0.20354906 -0.28702695  1.72952687  1.4289705 -0.1596993 -0.7161976
## 5 -0.09261896  0.02345825  0.51201583  0.1544345  0.4318039 -0.8674060
## 6  1.69952325  1.19231791 -0.98179754 -0.9567773 -0.6933918  0.4656891
##           X6         X7         X8          X9        X10
## 1 -0.8705750 -0.7448252 -0.4639697  0.62502272 -0.8149674
## 2  0.3532248 -0.5860332 -0.7964403  0.84868110 -0.1065119
## 3  0.4707434 -0.6588069 -0.7327518 -0.29429307  0.6588927
## 4 -0.7774007  0.2502145  0.5987052 -0.04428773  0.6247479
## 5 -0.9066908  0.8946086 -0.9700185  0.09082626  0.6102134
## 6 -0.7381794  0.8650175  0.4108119  0.75677429 -0.2281439
```

a)

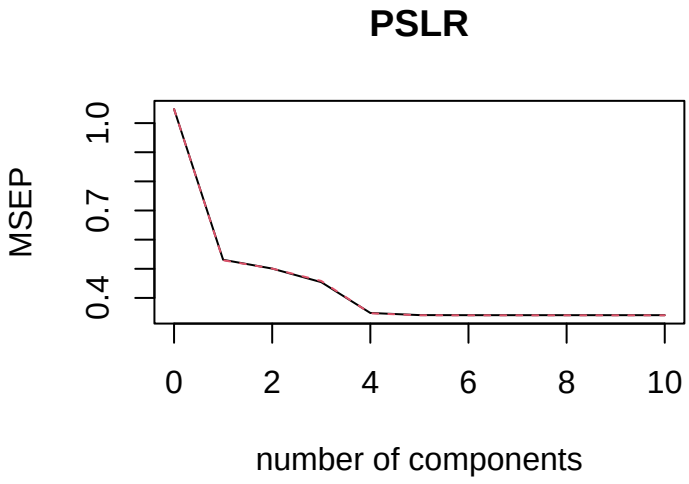
Fitting PCR and PSLR on the synthetic.train data set:

```
pcr_model <- pcr(Y~., data = synthetic.train, scale = TRUE, validation = "CV")
pslr_model <- plsr(Y~., data=synthetic.train, scale = TRUE, validation = "CV")
```

```
validationplot(pcr_model, val.type="MSEP", main="PCR")
```




```
validationplot(pslr_model, val.type="MSEP", main="PSLR")
```



b)

We can clearly see that the PSLR method shrinks the MSEP quicker for fewer components when comparing to the PCR method. PCR is a unsupervised method while PSLR is supervised.

Problem 3

a)

1.
2.
3. False (the extra term makes it less wiggly not more smooth?)
4. True (increase in k -> higher bias, lower variance ?)

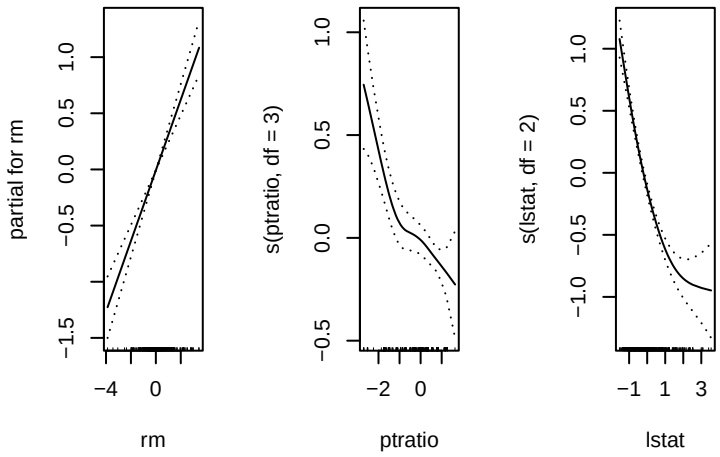
b)

using gam function to create a adaptive model:

```
adpt_mod <- gam(medv ~rm  + s(ptratio, df=3) + s(lstat, df=2), data=boston.train)
```

plotting results:

```
par(mfrow=c(1,3)) #to partition the Plotting Window
plot(adpt_mod, se=TRUE)
```



Problem 4

a)

1. False (?)
2. True
3. True
4. False

b)

c)

```
library(tidyverse)
library(palmerpenguins)  # Contains the data set 'penguins'.
data(penguins)
```

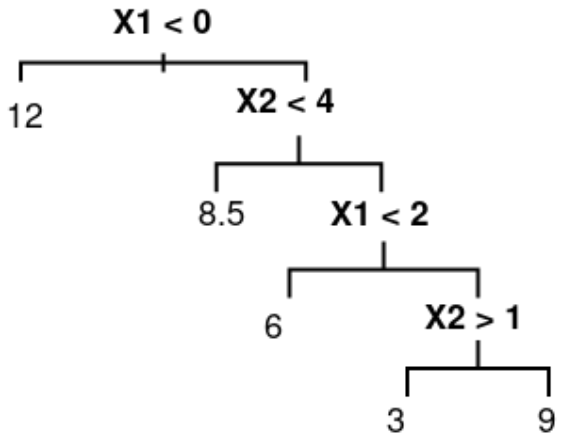


Figure 1: Sketch

```
names(penguins) <- c("species", "island", "billL", "billD", "flipperL", "mass", "sex",
    "year")

Penguins_reduced <- penguins %>% dplyr::mutate(mass = as.numeric(mass), flipperL = as.numeric(flipperL)
    year = as.numeric(year)) %>% drop_na()

# We do not want 'year' in the data (this will not help for future predictions)
Penguins_reduced <- Penguins_reduced[, -c(8)]

set.seed(4268)
# 70% of the sample size for training set
training_set_size <- floor(0.7 * nrow(Penguins_reduced))
train_ind <- sample(seq_len(nrow(Penguins_reduced)), size = training_set_size)
train <- Penguins_reduced[train_ind, ]
test <- Penguins_reduced[-train_ind, ]
```