Assignment 1

Group 2

2023-10-09

Task 1

Consider a balanced regression problem where for each input $x_i, i = 1, ..., N$, one has J repeated outputs $y_ij, j = 1, ..., J$ and one fits a parameterized model $f_\theta(x)$ by least squares.

- Show that the fit can be obtained from a least squares problem involving only xi and the average values $\overline{y_i} = \frac{1}{J}\sum_{j=1}^{J} y_{ij}$

The problem basically says that we have N groups, and for each we have J realizations y_{ij}. Therefore, we will indicate from now on the total number of observations as $K = N \cdot J$. In the first step we subtract and add the groups means. We thus define the RSS as the following:

$$
\begin{aligned}
&RSS = \sum_{k}^{K}(y_k - f_\theta(x_k))^2 = \\
&\sum_{k}^{K}(y_k - \overline{y_k} + \overline{y_k} - f_\theta(x_k))^2 = \\
&\sum_{k}^{K}(y_k - \overline{y_k})^2 + 2\sum_{k}^{K}(y_k - \overline{y_k})(\overline{y_k} - f_\theta(x_k)) + \sum_{k}^{K}(\overline{y_k} - f_\theta(x_k))^2 = \\
&\sum_{k}^{K}(y_k - \overline{y_k})^2 + 2\sum_{i}^{N}(\overline{y_i} - f_\theta(x_i))\sum_{j}^{J}(y_{ij} - \overline{y_i}) + \sum_{i}^{N}\sum_{j}^{J}(\overline{y_i} - f_\theta(x_i))^2
\end{aligned}
$$

Now we make two observations. The first is that in the first term we obtain a constant which does not depend on θ. Secondly, the term $\sum_j^J(y_j - \overline{y_k})$ equals 0, since they are distances from the mean. Hence, we remain with the last term, which is precisely what we want to show.

- Explain how the least squares problem changes if the design is not balanced, i.e., one has different number of repetitions for each input x_i

If we add the point that we have different number of observations for each group, then we have that:

$$RSS = \sum_{i}^{N} w_i(\overline{y_i} - f_\theta(x_i))^2$$

where w_i is the number of observations for each group i.

Task 2

Task 3

Task 4

Task 5

Task 6

Task 7

```r
# load data
data("diabetes", package = "lars")

# matrices can apparently be columns in df..
# good to know
y <- rnorm(100)
x1 = rnorm(100)
x2 = rnorm(100)
x <- cbind(x1, x2)
df <- data.frame(y = y, x = I(x))

# but this complicates things so we break up the
# structure and make a nice df
# function to get rid of Asis
unAsIs <- function(X) {
    if("AsIs" %in% class(X)) {
        class(X) <- class(X)[-match("AsIs", class(X))]
    }
    X
}

# extract y and x
y <- diabetes$y
x <- unAsIs(diabetes$x)

# make a new df with all the data
diabetes_df <- as.data.frame(cbind(y, x))
```

As instructed, we now set a seed and sample row indices from the set of integers running from 1 to the number of observations with equal probability.

```r
# set seed
set.seed(12)

# split the data into train and test
# step 1: sample 400 indices
ind <- sample(x = 1:nrow(diabetes_df), size = 400)

# subset the datasets as instructed
```

```
train <- diabetes_df[ind, ]
test <- diabetes_df[-ind, ]
```

The reason why random sampling is a good idea is that we are not really familiar with the dataset. Specifically we do not know whether observations were sorted by any of the variables available and if so we do not know at all by which one. Just taking the 400 first observations then would lead the information contained in training and test data to be biased by sorting leading ultimately to sampling bias in our estimations.

INSERT EXPLANATION ABOUT STANDARDIZED VARIABLES HERE.

To analyse the correlation structures we simply calculate a matrix with correlation of all columns in our dataset. The first column contains the correlations of the variables in X with y and the other columns and rows respectively contain the correlations between the columns in X. In general high absolute values of $corr(X_k, y)$ are desirable because this implies high co- or countermovement of the dependent and independent variables. This at least hints at predictive power of X_k, where k is the column index. Contrary, low values for $corr(X_k, X_j)$, $k \neq j$ are desirable as high values would introduce all the problems associated with multicollinearity, most prominently however the variance of the estimates will become inflated. This means nothing else than a loss in precision of estimates. Another huge problem is that multicollinearity is associated with “almost rank deficient” $X'X$ what can lead to problems if we run our model on a computer system.

```
library(kableExtra)
# exploration of correlation
correlation_matrix <- round(cor(train), 2)
# eliminate redundancies and make a nice table for the pdf
correlation_matrix[!lower.tri(correlation_matrix)] <- ""
kable(correlation_matrix, booktabs = T)
```

	y	age	sex	bmi	map	tc	ldl	hdl	tch	ltg	glu
y											
age	0.18										
sex	0.07	0.21									
bmi	0.6	0.17	0.09								
map	0.44	0.31	0.26	0.41							
tc	0.22	0.26	0.06	0.25	0.25						
ldl	0.19	0.23	0.16	0.26	0.21	0.9					
hdl	-0.4	-0.09	-0.39	-0.36	-0.21	0.04	-0.2				
tch	0.42	0.22	0.36	0.41	0.29	0.56	0.67	-0.74			
ltg	0.56	0.27	0.16	0.44	0.39	0.52	0.33	-0.4	0.61		
glu	0.37	0.29	0.24	0.38	0.4	0.33	0.28	-0.27	0.41	0.48	

```
# use training data to fit the full model
# get model formula from column names
f <- as.formula(paste0("y~", paste(colnames(train)[-1], collapse = "+")))

fit_full <- lm(data = train, formula = f)

# get variables significant at alpha = 0.05
summary_full <- summary(fit_full)
coefficients <- summary_full$coefficients
significant <- which(coefficients[, 4] < 0.05)[-1]

# in sample MSE
```

```r
MSE_full_in <- mean(fit_full$residuals^2)

# out of sample MSE
pred_full <- predict(fit_full, newdata = test)
MSE_full_out <- mean((test$y - pred_full)^2)

message(paste("In sample MSE is", MSE_full_in, sep = ": "))
```

```
## In sample MSE is: 2876.35849729569
```

```r
message(paste("Out of sample MSE is", MSE_full_out, sep = ": "))
```

```
## Out of sample MSE is: 2809.87939232709
```

```r
# use the significant variables only
f2 <- as.formula(paste0("y~", paste(colnames(train)[significant], collapse = "+")))

# estimate smaller model
fit_sig <- lm(data = train, formula = f2)

# in sample MSE
MSE_sig_in <- mean(fit_sig$residuals^2)

# out of sample MSE
pred_sig <- predict(fit_sig, newdata = test)
MSE_sig_out <- mean((test$y - pred_sig)^2)

message(paste("In sample MSE is", MSE_sig_in, sep = ": "))
```

```
## In sample MSE is: 3030.1546413137
```

```r
message(paste("Out of sample MSE is", MSE_sig_out, sep = ": "))
```

```
## Out of sample MSE is: 3222.23929484251
```

```r
# F-Test
anova(fit_full, fit_sig)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ age + sex + bmi + map + tc + ldl + hdl + tch + ltg + glu
## Model 2: y ~ sex + bmi + map + ltg
##   Res.Df     RSS Df Sum of Sq      F   Pr(>F)
## 1    389 1150543
## 2    395 1212062 -6    -61518 3.4666 0.002383 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Task 8

The Akaike criterion is in general given by

$$AIC = 2K - ln(L)$$

where K is the number of regressors and L is the likelihood of the model. Since $2K$ is a penalty term, AIC is to be minimized to find the most appropriate model.

Best Subset Selection

Best subset selection is basically just getting all 2^P possible combinations of explanatory variables available to us and estimating the corresponding regression models. It would be conventient to use the leaps package for this task but there the AIC-criterion is not implemented but only SIC and BIC. So we have to construct the models ourselves, estimate them and calculate AIC.

First we write a function that calculates all possible combinations of regressors and gives back the associated regression formulas. We will not use the leaps and bounds algorithm as the dataset is not to big.

```
# write function to get formulas
bs_formulas <- function(x = train, dep = "y", intercept_only = T) {
  # extract variables names
  vars <- names(x)
  exps <- vars[vars != dep]

  # get all combinations
 f <- lapply(1:length(exps), function(k) {
    # get combinations for given k
    combinations <- combn(exps, m = k, simplify = F)
    # make it a regression formula
    formulas <- lapply(combinations, function(c) {
      paste(dep, paste(c, collapse = "+"), sep = "~")
    })
    # make it a vector again
    unlist(formulas)
    })
  # dissolve list again
  output <- unlist(f)

  if(intercept_only == T) output <- c(as.formula("y ~ 1"), output)

  return(output)
}
```

Then we estimate the models

```
# get formulas
formulas <- bs_formulas() # look at defaults set above

# estimate all models
fits <- lapply(formulas, function(f) lm(data = train, formula = f))

# get log likelihoods
```

```
LL <- unlist(lapply(fits, function(f) logLik(f)[1]))

# get number of regressors (K)
K <- unlist(lapply(fits, function(fit) length(fit$coefficients)))

# get AIC
AIC <- 2 * K - 2 * LL

# get the most appropriate model
best_index <- which.min(AIC)

# display it
formulas[[best_index]]
```

```
## [1] "y~sex+bmi+map+tc+ldl+ltg"
```

Now that we have identified the best model we can assess its in- and out-of-sample performance using MSE again.

```
# get in sample MSE
MSE_BS_in <- mean(fits[[best_index]]$residuals^2)

# get out of sample MSE
MSE_BS_out <- mean((test$y - predict(fits[[best_index]], newdata = test))^2)
```

Finally we conduct an F-Test of the identified model against the full model.

```
anova(fits[[best_index]], fits[[length(fits)]]) # last model is the full one by construction
```

```
## Analysis of Variance Table
##
## Model 1: y ~ sex + bmi + map + tc + ldl + ltg
## Model 2: y ~ age + sex + bmi + map + tc + ldl + hdl + tch + ltg + glu
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1    393 1152326
## 2    389 1150543  4    1782.5 0.1507 0.9626
```

Backward Stepwise

Here we are lucky because the MASS package provides us with a function that does stepwise regression based on AIC.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.0.5
```

```
# conduct backwards stepwise regression
backwards_step <- stepAIC(fit_full, direction = "backward")
```

```
## Start:  AIC=3207.71
## y ~ age + sex + bmi + map + tc + ldl + hdl + tch + ltg + glu
##
##        Df Sum of Sq     RSS    AIC
## - age   1         6 1150550 3205.7
## - hdl   1        12 1150555 3205.7
## - tch   1       448 1150991 3205.9
## - glu   1      1011 1151555 3206.1
## - ldl   1      4542 1155086 3207.3
## <none>              1150543 3207.7
## - tc    1      7009 1157553 3208.1
## - sex   1     24865 1175408 3214.3
## - ltg   1     48634 1199178 3222.3
## - map   1     55404 1205947 3224.5
## - bmi   1    184098 1334642 3265.1
##
## Step:  AIC=3205.71
## y ~ sex + bmi + map + tc + ldl + hdl + tch + ltg + glu
##
##        Df Sum of Sq     RSS    AIC
## - hdl   1        11 1150560 3203.7
## - tch   1       448 1150998 3203.9
## - glu   1      1007 1151556 3204.1
## - ldl   1      4537 1155087 3205.3
## <none>              1150550 3205.7
## - tc    1      7003 1157553 3206.1
## - sex   1     25368 1175918 3212.4
## - ltg   1     48813 1199363 3220.3
## - map   1     56735 1207285 3223.0
## - bmi   1    184186 1334736 3263.1
##
## Step:  AIC=3203.72
## y ~ sex + bmi + map + tc + ldl + tch + ltg + glu
##
##        Df Sum of Sq     RSS    AIC
## - tch   1       653 1151214 3201.9
## - glu   1      1016 1151577 3202.1
## <none>              1150560 3203.7
## - ldl   1      9967 1160528 3205.2
## - sex   1     25494 1176055 3210.5
## - tc    1     27291 1177851 3211.1
## - map   1     56731 1207291 3221.0
## - ltg   1     99925 1250485 3235.0
## - bmi   1    184206 1334767 3261.1
##
## Step:  AIC=3201.95
## y ~ sex + bmi + map + tc + ldl + ltg + glu
##
##        Df Sum of Sq     RSS    AIC
## - glu   1      1112 1152326 3200.3
## <none>              1151214 3201.9
## - sex   1     24844 1176058 3208.5
## - ldl   1     31590 1182804 3210.8
## - tc    1     52789 1204003 3217.9
## - map   1     56116 1207330 3219.0
## - bmi   1    184285 1335499 3259.3
## - ltg   1    227787 1379001 3272.2
##
## Step:  AIC=3200.33
## y ~ sex + bmi + map + tc + ldl + ltg
##
##        Df Sum of Sq     RSS    AIC
## <none>               1152326 3200.3
## - sex   1     23911 1176237 3206.5
## - ldl   1     32030 1184356 3209.3
## - tc    1     52700 1205026 3216.2
## - map   1     60846 1213172 3218.9
## - bmi   1    192497 1344823 3260.1
## - ltg   1    248962 1401288 3276.6
```

```
# get in sample MSE
MSE_backwards_in <- mean(backwards_step$residuals^2)

# get out of sample MSE

MSE_backwards_out <- mean((test$y - predict(backwards_step, newdata = test))^2)

anova(fit_full, backwards_step)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ age + sex + bmi + map + tc + ldl + hdl + tch + ltg + glu
## Model 2: y ~ sex + bmi + map + tc + ldl + ltg
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1    389 1150543
## 2    393 1152326 -4   -1782.5 0.1507 0.9626
```

Task 9

Task 10