Applied Statistical Methods - Solution 6

Peter von Rohr

2021-03-29

Problem 1: Example with LASSO

The file available at https://charlotte-ngs.github.io/gelasmss2021/data/asm_ex06_p01_lasso.txt contains a dataset with genotypes from 100 SNP-Loci. In addition to the genomic information, the dataset also holds observations of a certain trait for 50 animals. The dataset can be read into a matrix in R with the following statement.

```
mat_lasso_data <- matrix(scan("https://charlotte-ngs.github.io/gelasmss2021/data/asm_ex06_p01_lasso.txt"), nrow = 50, byrow = TRUE)
```

Let us have a look at the first 5 rows and the first 5 columns of the matrix that stores the dataset.

```
mat_lasso_data[1:n_nr_row,1:n_nr_col]
```

```
##           [,1] [,2] [,3] [,4] [,5]
## [1,] -40.39872   -1    0   -1   -1
## [2,] -46.35871   -1   -1   -1    0
## [3,] -33.60278   -1   -1   -1   -1
## [4,] -48.47177    0   -1   -1   -1
## [5,] -38.82089   -1   -1    0   -1
```

From this output, we can see that the observations of all animals can be found in the first column of the data matrix `mat_lasso_data`. In columns 2 to 101 of the data matrix there are the genotypes of all SNP loci. The linear model is fitted with LASSO using the function `glmnet()` from the package `glmnet`. The SNP genotypes are used as explanatory variables and the observations are the response variables.

Your Tasks

- Use the following R-Statement to estimate the SNP-effects using LASSO

```
require(glmnet)
fitsnp <- glmnet(x = mat_lasso_data[, -1], y = mat_lasso_data[, 1])
```

- Visualize the dependency between the value of the penalty term λ and the number of explanatory variables which are not 0.

```
plot(fitsnp, xvar = "lambda", label = TRUE)
```

- Use a cross-validation to determine the value of λ.

```
cvfitsnp <- cv.glmnet(x = mat_lasso_data[, -1], y = mat_lasso_data[, 1])
```

- Show the results of the cross-validation in a plot using the function `plot()`.

```
plot(cvfitsnp)
```

- In the plot of the cross-validation results there are two dashed lines which both indicated special values for λ. The first value is the minimum of all λ-values and the second is the one that sets the most explanatory variables to 0 with the restriction that the sum of squared errors is not further away than one standard deviation from its minimum. The two λ-values are obtained with

```
cvfitsnp$lambda.min
cvfitsnp$lambda.1se
```

- Find all coefficients which are not 0 for both λ-values and compare them to the true values taken from the simulation.

```
coefmin <- coef(cvfitsnp, s = "lambda.min")
(cofminnz <- coefmin[coefmin[, 1] != 0,])
```

```
coef1se <- coef(cvfitsnp, s = "lambda.1se")
(coef1senz <- coef1se[coef1se[, 1] != 0, ])
```

The true SNP-positions from the simulation are:

```
(vec_sign_snp_idx <- c(73,54,26,30,7))
```

```
## [1] 73 54 26 30  7
```

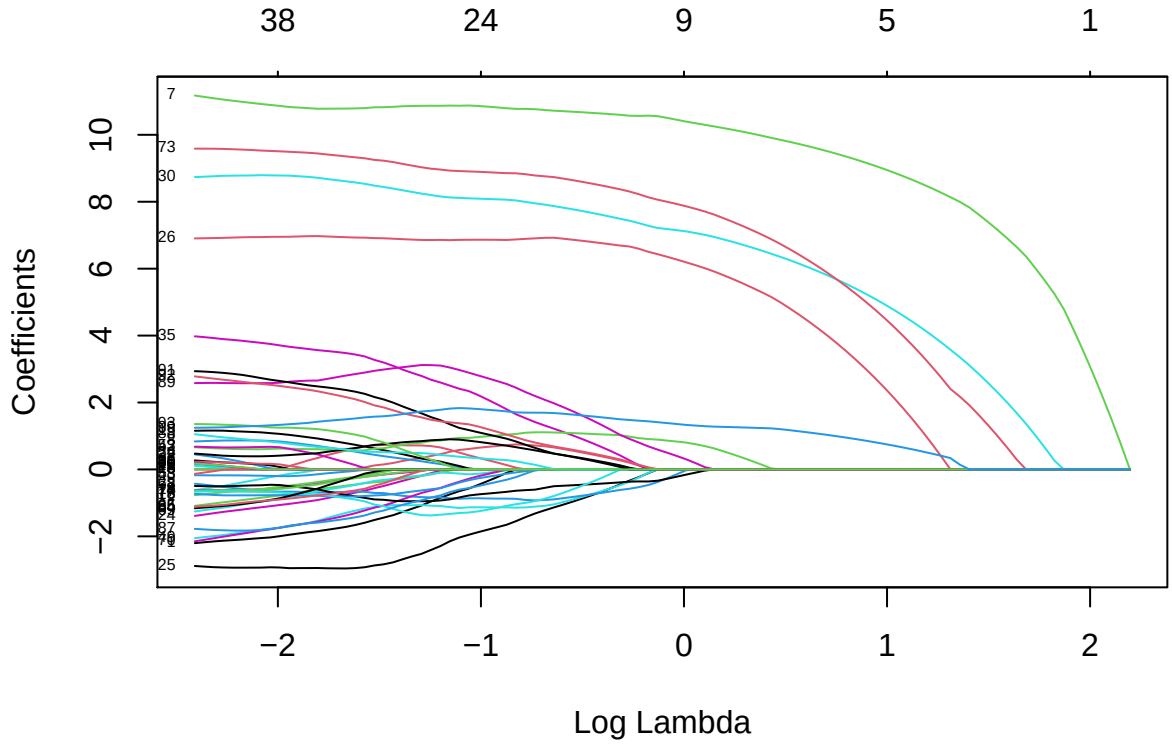
Solution

The linear model is fitted with the following statement.

```
require(glmnet)
fitsnp <- glmnet(x = mat_lasso_data[, -1], y = mat_lasso_data[, 1])
```

The result of the model fit is a `glmnet`-object. The resuling object is best viewed with the following plot.

```
plot(fitsnp, xvar = "lambda", label = TRUE)
```

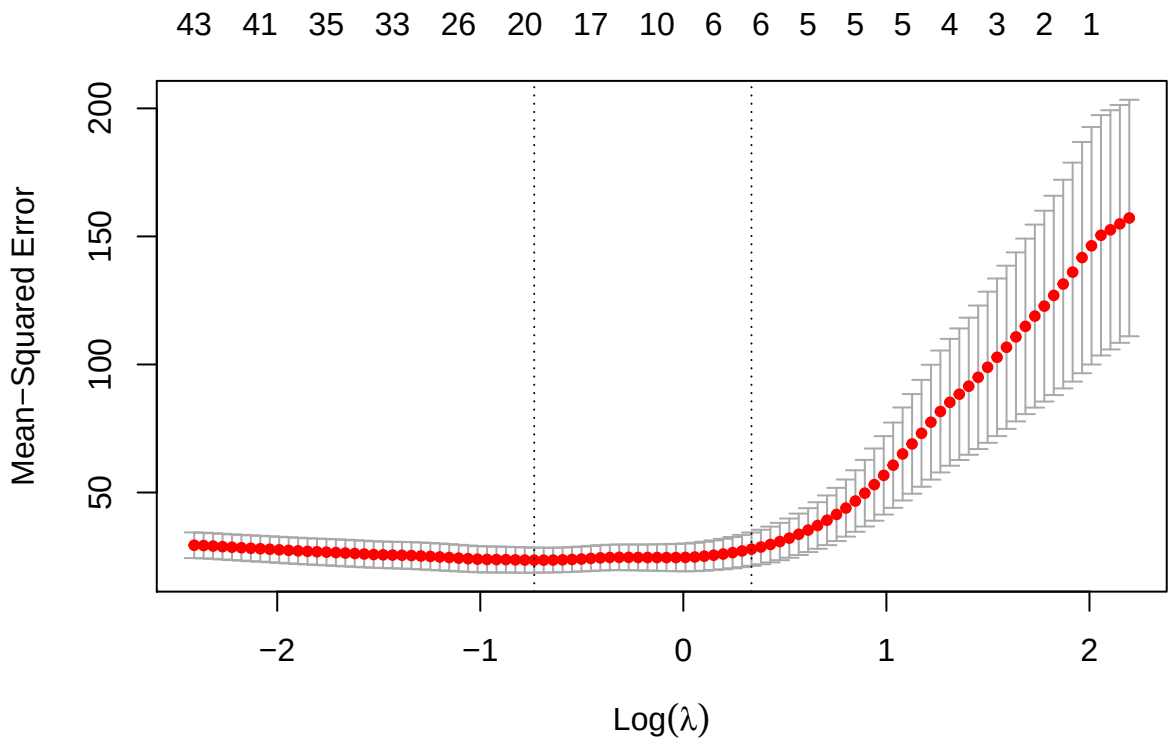


The cross-validation is done with the function `cv.glmnet()`.

```
cvfitsnp <- cv.glmnet(x = mat_lasso_data[, -1], y = mat_lasso_data[, 1])
```

The results can be shown with the following plot.

```
plot(cvfitsnp)
```



The special λ-values indicated with the dashed lines can be obtained with

```
cvfitsnp$lambda.min
```

```
## [1] 0.4798521
```

```
cvfitsnp$lambda.1se
```

```
## [1] 1.398794
```

The coefficients corresponding to the SNP-positions which are not 0 for both λ-values are obtained with

```
coefmin <- coef(cvfitsnp, s = "lambda.min")
(cofminnz <- coefmin[coefmin[, 1] != 0,])
```

```
##  (Intercept)           V5           V7          V25          V26          V30          V35
## -15.79398265   0.71636758  10.76559820  -1.31889450   6.90504960   7.95290843   1.41726657
##          V49          V54          V68          V72          V73          V76          V77
##  -1.11112785   0.57060907  -0.92522305  -0.60425660   8.78496375  -0.02648076  -0.94782017
##          V88          V89          V91          V92          V99
##   0.13929355   2.19675736   0.64615444   0.80421694   1.69646734
```

```
coef1se <- coef(cvfitsnp, s = "lambda.1se")
(coef1senz <- coef1se[coef1se[, 1] != 0, ])
```

```
## (Intercept)          V7         V26         V30         V42         V73         V99
## -21.6517234  10.0349793   5.4325943   6.6300267   0.2493211   7.1502830   1.2459546
```

The SNP-positions are extracted from the coefficients as follows

```
(s_snp_pos_min <- gsub(pattern = "V", replacement = "",
                       setdiff(names(cofminnz), "(Intercept)"),
                       fixed = TRUE))
```

```
##  [1] "5"  "7"  "25" "26" "30" "35" "42" "49" "54" "68" "72" "73" "76" "77" "87" "88" "89" "91" "92"
```

The match between the estimated and the true SNP-positons using minimal λ are

```
(vec_match_snp_min <- intersect(s_snp_pos_min, as.character(vec_sign_snp_idx)))
```

```
## [1] "7"  "26" "30" "54" "73"
```

```
(s_snp_pos_1senz <- gsub(pattern = "V", replacement = "",
                         setdiff(names(coef1senz), "(Intercept)"),
                         fixed = TRUE))
```

```
## [1] "7"  "26" "30" "42" "73" "99"
```

The match between the estimated and the true SNP-positons using the SE-λ are

```
(vec_match_snp_1senz <- intersect(s_snp_pos_1senz, as.character(vec_sign_snp_idx)))
```

```
## [1] "7"  "26" "30" "73"
```

Problem 2: Bayesian Regression Analysis

Given is the earlier used dataset of `breast circumference` and `body weight`.

Table 1: Dataset for Regression of Body Weight on Breast Circumference for ten Animals

Animal	Breast Circumference	Body Weight
1	176	471
2	177	463
3	178	481
4	179	470
5	179	496
6	180	491
7	181	518
8	182	511
9	183	510
10	184	541

The model that is used is a simple linear regression model given by

$$y_i = \beta_0 + \beta_1 * x_i + \epsilon_i$$

.

where y_i corresponds to the body weight of animal i, x_i is the breast circumference of animal i, β_0 is the unknown intercept and β_1 is the unknown regression coefficient. For reasons of simplicity, we assume the residual variance σ^2 to be known. For the later computations, we insert the estimate that is obtained from the `lm()` function. This value corresponds to $\sigma^2 = 122.8$.

Bayesian Estimation Of Unknowns

As already mentioned during the lecture, Bayesian estimates of unknowns are based on the posterior distribution of the unknowns given the knowns. For our regression model the unknowns correspond to

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

The posterior distribution of the unknowns given the knowns is $f(\beta|y)$. Using Bayes' Theorem we can write $f(\beta|y)$ as

$$\begin{aligned} f(\beta|y) &= \frac{f(\beta, y)}{f(y)} \\ &= \frac{f(y|\beta)f(\beta)}{f(y)} \\ &\propto f(y|\beta)f(\beta) \end{aligned}$$

When we do not have any specific prior knowledge about β, the prior distribution $f(\beta)$ for the unknown β is set to a constant. Therefore we can write

$$\begin{aligned} f(\beta|y) &\propto f(y|\beta)f(\beta) \\ &\propto f(y|\beta) \end{aligned}$$

Assuming a normal distribution for the data causes the likelihood $f(y|\beta)$ to be a multivariate normal distribution.

$$\begin{aligned} f(\beta|y) &\propto f(y|\beta) \\ &= (2\pi\sigma^2)^{-n/2} exp\left\{-\frac{1}{2}\frac{(y - X\beta)^T(y - X\beta)}{\sigma^2}\right\} \end{aligned} \tag{1}$$

The above expression (1) is an $n-$ dimensional normal distribution with expected value $X\beta$ and variance-covariance matrix corresponding to $I\sigma^2$. But because we have just two unknowns β_0 and β_1 the posterior distribution $f(\beta|y)$ must have two dimensions and not n. The following re-arrangement can solve this problem. Let us set the variable Q to

$$Q = (y - X\beta)^T(y - X\beta) = y^T y - 2y^T X\beta + \beta^T(X^T X)\beta$$

Introducing the least squares estimate $\hat{\beta} = (X^T X)^{-1}X^T y$ into the above equation by replacing $y^T X$ with $\hat{\beta}^T(X^T X)$ results in

$$Q = y^T y - 2\hat{\beta}^T(X^T X)\beta + \beta^T(X^T X)\beta = y^T y + (\beta - \hat{\beta})^T(X^T X)(\beta - \hat{\beta}) - \hat{\beta}^T(X^T X)\hat{\beta}$$

Inserting this last result back into (1) gives

$$
\begin{aligned}
f(\beta|y) &\propto f(y|\beta) \\
&= (2\pi\sigma^2)^{-n/2} exp\left\{-\frac{1}{2}\frac{(y-X\beta)^T(y-X\beta)}{\sigma^2}\right\} \\
&= (2\pi\sigma^2)^{-n/2} exp\left\{-\frac{1}{2}\frac{y^Ty+(\beta-\hat{\beta})^T(X^TX)(\beta-\hat{\beta})-\hat{\beta}^T(X^TX)\hat{\beta}}{\sigma^2}\right\} \\
&= (2\pi\sigma^2)^{-n/2}\left[exp\left\{-\frac{1}{2}\frac{y^Ty}{\sigma^2}\right\} * exp\left\{-\frac{1}{2}\frac{(\beta-\hat{\beta})^T(X^TX)(\beta-\hat{\beta})}{\sigma^2}\right\} * exp\left\{-\frac{1}{2}\frac{-\hat{\beta}^T(X^TX)\hat{\beta}}{\sigma^2}\right\}\right] \\
&\propto exp\left\{-\frac{1}{2}\frac{(\beta-\hat{\beta})^T(X^TX)(\beta-\hat{\beta})}{\sigma^2}\right\} \qquad (2)
\end{aligned}
$$

The last proportionality results from the fact that only the term depending on β is retained. All other terms not depending on β are constant factors with respect to β and can therefore be dropped. Thus $f(\beta|y)$ can be written as

$$
f(\beta|y) \propto exp\left\{-\frac{1}{2}\frac{(\beta-\hat{\beta})^T(X^TX)(\beta-\hat{\beta})}{\sigma^2}\right\}
$$

which is recognized as proportional to a two dimensional normal density with mean $\hat{\beta}$ and variance $(X^TX)^{-1}\sigma^2$. Thus in the simple setting the mean of the posterior mean can already be seen from the above formula. But in a more complex setting, the posterior distribution does not have a standard form and we need to setup a sampling scheme which allows us to draw random numbers from the posterior distribution. The sampling scheme that we are introducing here is called the **Gibbs Sampler**.

Gibbs Sampler for β

The simple regression model that we are using for the breast circumference and the body weight data can be written in matrix-vector notation as

$$
y = 1\beta_0 + x\beta_1 + \epsilon
$$

In the Gibbs sampling scheme both unknowns β_0 and β_1 are sampled from their full conditional distributions. For β_0 the full conditional posterior distribution is $f(\beta_0|\beta_1, y)$ which is computed for the current value of β_1. Separating β_0 from the other unknowns yields the linear model

$$
w_0 = 1\beta_0 + \epsilon
$$

where $w_0 = y - x\beta_1$. The least squares estimator of β_0 is

$$
\hat{\beta}_0 = (1^T1)^{-1}1^Tw_0
$$

with variance

$$
var(\hat{\beta}_0) = (1^T1)^{-1}\sigma^2
$$

Applying the same strategy as for $f(\beta|y)$, it can be shown that $f(\beta_0|\beta_1, y)$ is a normal distribution with mean $\hat{\beta}_0$ as mean and $(1^T1)^{-1}\sigma^2$ as variance. The full-conditional posterior of β_1 can be derived the same way, leading to

$$\hat{\beta}_1 = (x^T x)^{-1} x^T w_1$$

with variance $var(\hat{\beta}_1) = (x^T x)^{-1}\sigma^2$ where $w_1 = y - 1\beta_0$.

Your Task

- Create a Gibbs Sampling scheme for the dataset shown in Table 1.
- Use the mean of the generated samples as an estimate for the unknowns β_0 and β_1.

Solution

We have mentioned earlier that we are using the residual variance that is obtained from the least squares analysis which is shown just below.

```
lm_reg_bwbc <- lm(`Body Weight` ~ `Breast Circumference`, data = tbl_reg)
n_res_var <- sum(lm_reg_bwbc$residuals^2)/lm_reg_bwbc$df.residual
n_res_var_rounded <- round(n_res_var, digits = 1)
```

We start by setting up the matrix X with two columns. The first column contains only ones and the second column contains the measured breast circumference data.

```
n_nr_obs <- nrow(tbl_reg)
X <- matrix(c(rep(1, n_nr_obs), tbl_reg$`Breast Circumference`), ncol = 2)
```

In the next step, we assign the observed body weights to the vector y.

```
y <- tbl_reg$`Body Weight`
```

Before, the sampling iterations are started, the vector used for the sampled values is initialised.

```
beta <- c(0,0)
meanBeta <- c(0,0)
```

The random samples are drawn in a loop where in turn the unknowns are updated. Before starting the loop, we have to fix the random number generator seed, such that we get reproducible results.

```
#' fix the seed
set.seed(123942)
#' fix the number of samples
niter <- 100000
#' loop over the iterations
for (iter in 1:niter){
  # sampling the intercept beta_0
  w <- y - X[,2] * beta[2]
  x <- X[,1]
  xtxi <- 1/crossprod(x)
  betaHat <- crossprod(x, w) * xtxi
  beta[1] <- rnorm(1, betaHat, sqrt(xtxi * n_res_var))
  # sample the slope beta_1
  w <- y - X[,1] * beta[1]
  x <- X[,2]
  xtxi <- 1/crossprod(x)
  betaHat <- crossprod(x, w) * xtxi
  beta[2] <- rnorm(1, betaHat, sqrt(xtxi * n_res_var))
  # sum up the estimate
```

```
  meanBeta <- meanBeta + beta
  # output every 10000 rounds
  if ((iter%%10000) == 0){
    cat(sprintf("Iteration: %d \n", iter))
    cat(sprintf("Intercept: %6.3f \n", meanBeta[1]/iter))
    cat(sprintf("Slope:     %6.3f \n", meanBeta[2]/iter))
  }
}
```

```
## Iteration: 10000
## Intercept: -294.094
## Slope:      4.388
## Iteration: 20000
## Intercept: -550.415
## Slope:      5.813
## Iteration: 30000
## Intercept: -719.016
## Slope:      6.750
## Iteration: 40000
## Intercept: -804.818
## Slope:      7.227
## Iteration: 50000
## Intercept: -887.182
## Slope:      7.684
## Iteration: 60000
## Intercept: -945.612
## Slope:      8.009
## Iteration: 70000
## Intercept: -945.389
## Slope:      8.008
## Iteration: 80000
## Intercept: -934.225
## Slope:      7.946
## Iteration: 90000
## Intercept: -932.951
## Slope:      7.939
## Iteration: 100000
## Intercept: -953.116
## Slope:      8.051
```

The last line of the above output corresponds to the Bayesian estimate of the intercept and the slope. That would need to be compared to the least squares estimate which is obtained from

```
summary(lm_reg_bwbc)
```

```
##
## Call:
## lm(formula = `Body Weight` ~ `Breast Circumference`, data = tbl_reg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.3941  -6.5525  -0.0673   9.3707  13.2594
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)            -1065.115    255.483  -4.169 0.003126 **
## `Breast Circumference`     8.673      1.420   6.108 0.000287 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.08 on 8 degrees of freedom
## Multiple R-squared:  0.8234, Adjusted R-squared:  0.8014
## F-statistic: 37.31 on 1 and 8 DF,  p-value: 0.000287
```