Exercise Series 11

1. **a)** Let's consider the general linear regression model:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}.$$

Show that this model is equivalent to the following one:

$$y_i - \bar{y} = \sum_{j=1}^{p} \beta_j \cdot (x_{ij} - \bar{x}_{\cdot j}).$$

Therefore by centering the variables it is always possible to get rid of the intercept β_0.

b) Show that the ridge-regression solution defined as

$$\tilde{\beta}^*(s) = \underset{\|\beta\|^2 \le s}{\arg\min} \|\mathbf{Y} - X\beta\|^{\mathbf{2}}$$

is given by

$$\hat{\beta}^*(\lambda) = (X^\intercal X + \lambda I)^{-1} X^\intercal \mathbf{Y}.$$

where λ is a suitably chosen Lagrange-multiplicator. Therefore the ridge estimator is still linearly depending on the response $\mathbf{Y}$. Note that for $\lambda > 0$ the ridge solution exists even if $X^\intercal X$ has not full rank. Therefore ridge regression is practicable also if $n \ll p$.

c) The *ridge traces* $\hat{\beta}^*(\lambda)$ can computationally easily be determined by using a *singular value decomposition* of the data matrix $X = UDV^\intercal$ where $U(n \times p)$ and $V(p \times p)$ are orthogonal and D is diagonal. Show that:

$$\hat{\beta}^*(\lambda) = V(D^2 + \lambda I)^{-1} D U^\intercal Y.$$

d) Show that the ridge regression fit is just a linear combination of shrinked response-components y_i with respect to the orthogonal basis defined by U. More explicitly show that:

$$\hat{y}_{ridge}(\lambda) = \sum_{j=1}^{p} \mathbf{u_j} \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u_j}^\intercal \mathbf{y},$$

where d_j are the diagonal elements of D. In fact one can show that the directions defined by $\mathbf{u_j}$ are the so called *principal components* of the dataset X. The smaller the corresponding d_j-value, the smaller the data variance in direction u_j. For directions with small data variance, the gradient estimation for the minimization problem is difficult, therefore ridge regression shrinks the corresponding coefficients the most.

e) Ridge regression can also be motivated by Bayesian theory. We assume that

$$\mathbf{Y}|\beta \sim \mathcal{N}(X\beta, \sigma^{\mathbf{2}} I) \text{ and } \beta \sim \mathcal{N}(\mathbf{0}, \tau I).$$

Show that the ridge estimator $\hat{\beta}^*(\lambda)$ is the mean of the posterior distribution. What is the relationship between λ, τ and σ^2?

2. Once again we look at the dataset `vehicle.dat` which still can be found at:

`"http://www.ethz.ch/Teaching/Datasets/NDK/vehicle.dat"`

This time we apply a shrinking regression method as our classifier, namely the *lasso*, which does variable selection. We now want to include interaction terms between variables. To do this, we transform the dataset `vehicle.dat` in the following way:

```
mf.veh <- model.frame(Class ~ .^2, data =d.vehicle)
Y.d2 <- model.response(mf.veh)
X.d2 <- model.matrix(Class ~ .^2 - 1, mf.veh) # without Intercept (!!!)
```

The model matrix `X.d2` contains now, besides the original predictors, the products of each two of them.

The aim of this exercise is to find the optimal degree of shrinking by cross-validation and the one-standard-error-rule.

Lasso is provided by function `lars()` in the homonymous package `lars`.

a) Because we use a plain non-generalized regression method as classifier in a mutlticlass-classification problem (remember that the `Class`-variable consists of *four* factors `bus`,`van`, `saab`,`opel`) we can choose a *one against the rest*- approach (as described in the lecture notes in section 6.4.2). Write a function `cl.lasso` which calculates the misclassification rate. The `input` should consist of a vector containig different values for the penalization parameter, an arbitrary training dataset (to fit the model) and an arbitrary test dataset (to evaluate the missclassification rate). The `output` should be a vector containing the misclassification rates for every value of the s-parameter.

R-Hints:

For `lars`-objects there are methods `predict()` and `coef()` which can be used for prediction. See `?predict.lars` for details. Use the option `mode = "fraction"` for `predict`. Then the tuning parameter `s` can be interpreted nicely as it corresponds to a regression coefficient whose L_1-norm is s% of the corresponding least-squares coefficient vector's L_1-norm. Therefore a convenient choice for s is:

```
s <- seq(0,1,length=101)
```

Because `predict()` of a lars-object gives the regression coefficients for *all* values of the tuning parameter vector `s` at once, it is convenient to store the probabilites in a 3-dimensional array `prob`, where the first index is over the datapoints in the test set, the second over the class levels and the third over the components of `lambda`.

```
prob[,i,]  <- predict(m.lasso,newx=x.new,s=s,type="fit",
                mode="fraction")$fit
```

The prediction can then be made with

```
pred <- apply(prob,3,max.col)  ,
```

where `pred` is now a matrix containing the prediction (1 for the first class, 2 for the second class and so on) for every datapoint in the test set and for every s-value.

b) Write a K-fold-CV-function to determine optimal values for the tuning parameter `s`. Choose the misclassification-error as your CV-criterion, i.e., use the function from part a) in your code. Applying the one-standard-error-rule, all s-values with a CV score below the minimal CV score plus one standard error are reasonable. From these values, we choose the smallest one, because the smaller the s-value the less variables are selected ($s = 0$ forces all coefficients to be zero, which means that no variable is selected). The standard error can be calculated via the formula

$$\hat{s.e.} = \left(\frac{1}{K(K-1)S} \sum_{j=1}^{S} \sum_{i=1}^{K} (me_i(s_j) - \bar{me}(s_j))^2 \right)^{\frac{1}{2}},$$

where $me_i(s_j)$ is the missclassification rate for the j-th s-value using the i-th block from the CV-partition as test set, $\bar{me}(s_j)$ is the average missclassification rate at the j-th s value over all K blocks and S is the number of s-values. The "optimal" s-value s_{opt} is then

$$s_{opt} = \min_j \{s_j : \bar{me}(s_j) \leq \min_j \{\bar{me}(s_j)\} + \hat{s.e.}\}.$$

Write the CV-function in such a way that it returns the average missclassification rate $\bar{me}(s_j)$ for every $j \in \{1, 2, \ldots, S\}$ (as a vector) and the standard error $\hat{s.e.}$.
For your computation, choose $K = 20$, plot $\bar{me}(s_j)$ $j = 1, 2, \ldots, S$ versus s_j $j = 1, 2, \ldots, S$ and add the one-standard-error-rule threshhold $\min_j\{\bar{me}(s_j)\} + \hat{s.e.}$ to the picture.

c) Plot the lasso-traces (for the *whole* data-set) and fit the optimal model .
As already mentioned, the lasso can be used for *variable selection*, because of L_1-penalization many of the fitted lasso-method regression coefficients become 0. Find out, how many and which variables are selected in each of the 4 one-against-the-others classifications and in total, i.e., for all 4.
R-Hints: For traces-plotting you can use the ordinary `plot`-function.
To find the selected variables of a `lars`-fit and their number (for a fixed s-value ss), you might want to use the following code:

```
fit <- lars(x=???,y=???,trace=FALSE)
is.cf.n0 <- coef(fit,s=ss,mode="fraction")!=0
sel <- colnames(X.d2)[is.cf.n0] ## selected variable names
nsel <- length(sel) ## number of selected variables
```

Preliminary discussion: Friday, May 23, 2008.
Deadline: Friday, May 30, 2008.
Notice: In this serie you can get a bonus point (3rd point) if you solve 2b) and 2c) additionally.