Regression - II

Prof. Dan A. Simovici

UMB

When the number n of input variables is large, the linear independence of the columns $\mathbf{b}^1, \ldots, \mathbf{b}^n$ of the design matrix B may not hold and the rank of B may be smaller than n.
The linear dependencies that may exist between the columns of B (reflecting linear dependencies among experiment variables) invalidate the assumptions previously made. These dependencies are known as *colinearities* among variables.

- One solution is to replace $B'B$ in the least-square estimate $\hat{\mathbf{r}} = (B'B)^{-1}B'\mathbf{y}$ by $B'B + \lambda I_n$ and to define the *ridge regression estimate* as $\mathbf{r}(\lambda) = (B'B + \lambda I_n)^{-1}B'\mathbf{y}$.
- The term *ridge regression* is justified by the fact that the main diagonal in the correlation matrix may be thought of as a ridge of this matrix.
- We retrieve the ridge regression estimate as a solution of *a regularized optimization problem*, that is, as an optimization problem where the objective function is modified by adding a term that has an effect the shrinking of regression coefficients.

Instead of minimizing the function $f(\mathbf{r}) = \| B\mathbf{r} - \mathbf{y} \|_2$ we use the objective function

$$g(\mathbf{r}, \lambda) = \| B\mathbf{r} - \mathbf{y} \|_2 + \lambda \| \mathbf{r} \|^2 .$$

This approach is known as *Tikhonov regularization method* and g is known as the *ridge loss function*.

A necessary condition of optimality is $(\nabla g)_{\mathbf{r}} = \mathbf{0}_n$. This yields:

$$
\begin{aligned}
(\nabla g)_{\mathbf{r}} &= 2B'B\mathbf{r} - 2B'\mathbf{y} + 2\lambda\mathbf{r} \\
&= 2(B'B\mathbf{r} - B'\mathbf{y} + \lambda\mathbf{r}) \\
&= 2[(B'B + \lambda I_n)\mathbf{r} - B'\mathbf{y}] = \mathbf{0}_n,
\end{aligned}
$$

which yields the previous estimate of $\mathbf{r}$. The ridge estimator is therefore a stationary point of g.

The Hessian of g is the matrix $H_g(\mathbf{x}) = \left(\frac{\partial^2 f}{\partial r_j \, \partial r_k}\right)$, and it is easy to see that

$$H_g(\mathbf{x}) = 2(B'B + \lambda I_n).$$

This implies that H_g is positive definite, hence the stationary point is a minimum.
Note that the ridge loss function is convex, as a sum of two convex functions. Therefore, the stationary point mentioned above is a global minimum of this function.

There are several **R** packages that need to be downloaded and installed in order to run the next example:

- `magrittr` which ensures that we can pipe results of an operations into another operation using the piping operator %>%;
- `dplyr` that allows us add new variables that are functions of existing variables. using the function `mutate()`, to pick variables based on their names using the function `select()`, or to extract cases based on their values using the function `filter()`;
- `glmnet` that provides the function `glmnet()` to be used for ridge regression. To achieve ridge regression we must specify the parameter `alpha = 0`.

Note that:

- ridge regression requires a vector input and a matrix of predictors rather than a formula and a data frame;
- ridge regressioin involves tuning a hyperparameter λ, and `glmnet` generates default values;
- It is possible to define our own value for λ.

An example using the mtcars data set

The data set `mtcars` is part of the basic **R** :

```
> data(mtcars)
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ..
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Add packages (I am assumming that you downloaded these already):

```
> library(magrittr)
> library(dplyr)
> library(glmnet)
```

Next, we build the vector `y` and the matrix of predictors `x`:

```
> y <- mtcars$hp
> x <- mtcars %>% select(mpg,wt,drat) %>% data.matrix()
```

The range of `lambda` is specified in the sequence `lambdas`:

```
> lambdas <- 10^seq(3,-2,by = -.1)
```

The variable y that is to be predicted is the power (hp) dependent on the variable included in the data frame x that includes the variables mpg, wt, and drat, representing the miles per galon, weight, and rear axle ratio, respectively.
The process will result in a prediction for y based on the variables that participate in the data frame x.

The construction of the ridge regression model is achieved with

```
> fit <- glmnet(x,y,alpha=0,lambda = lambdas)
> summary(fit)
          Length Class     Mode
a0         51    -none-    numeric
beta      153    dgCMatrix S4
df         51    -none-    numeric
dim         2    -none-    numeric
lambda     51    -none-    numeric
dev.ratio  51    -none-    numeric
nulldev     1    -none-    numeric
npasses     1    -none-    numeric
jerr        1    -none-    numeric
offset      1    -none-    logical
call        5    -none-    call
nobs        1    -none-    numeric
```

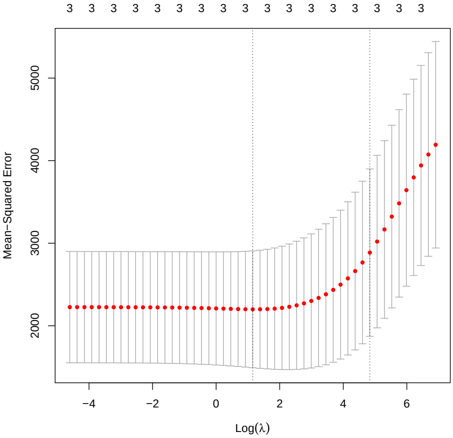
The function `glmnet` runs the model many times for different values of lambda, as specified by the sequence lambdas. This is accomplished with the statement

```
> cv_fit <- cv.glmnet(x,y,alpha=0,lambda = lambdas)
```

The results will be saved in a file named `plotcvfit.pdf` using the sequence

```
> pdf("plotcvfit.pdf")
> plot(cv_fit)
> dev.off()
```

This will produce the file `plotcvfit.pdf` that will be stored in your current directory and can be integrated afterwards in any LaTeX document. The graph is shown in the next slide.


3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
Mean−Squared Error
2000
3000
4000
5000
−4
−2
0
2
4
6
Log(λ)

The optimal value of lambda is determined writing

```
> opt_lambda <- cv_fit$lambda.min
> opt_lambda
[1] 3.162278
```

```
> fit <- cv_fit$glmnet.fit
> summary(fit)
           Length Class     Mode
a0          51    -none-    numeric
beta       153    dgCMatrix S4
df          51    -none-    numeric
dim          2    -none-    numeric
lambda      51    -none-    numeric
dev.ratio   51    -none-    numeric
nulldev      1    -none-    numeric
npasses      1    -none-    numeric
jerr         1    -none-    numeric
offset       1    -none-    logical
call         5    -none-    call
nobs         1    -none-    numeric
> y_predicted <- predict(fit,s=opt_lambda,newx=x)
> sst <- sum((y - mean(y))^2)
> sse <- sum((y_predicted - y)^2)
> rsq <- 1-sse/sst
```

- Ridge regression is less prone to overfitting training data. It might predict training data less well than the usual regression but it generalizes better to new data.
- This property of ridge regression is especially useful when variance in the training data is high (e.g., when the sample size is low and the number of features (variables) is high).