Ridge Regression and Lasso

1. Credit dataset

The data can be downloaded here http://www-bcf.usc.edu/~gareth/ISL/data.html, and imported into R using the `read.csv` function. This dataset is used in Chapter 6 in *An Introduction to Statistical Learning* by G. James, D. Witten, T. Hastie and R. Tibshirani to illustrate regularization techniques.

The learning sample contains 400 observations, for 10 variables. The response variable is `Balance`, which gives the average credit card debt for a number of individuals. The predictors are both quantitative, such as `Age`, `Cards` (number of credit cards), `Education` (number of years of education), `Income` (in thousands of dollars), `Limit` (credit limit), `Rating` (credit rating), and qualitative, `Gender`, `Student`, `Status` (marital status) and `Ethnicity` (Caucasian, African American or Asian). The variable `X` denotes the row number, and can be discarded. At this stage, you should reflect on ethical values of machine learning. Shall we be incorporating `Ethnicity` and `Gender` as predictors? Fairness in machine learning should not be discarded, as it is know that in many cases a poor use of algorithms or of the data used to train them increase inequalities amongst people, targeting minorities. I strongly suggest you read the book called *Weapons of Maths Destruction* by Cathy O'Neil, and listen to her TED talk https://www.ted.com/talks/cathy_o_neil_the_era_of_blind_faith_in_big_data_must_end?language=en. While we are discussing this, why not have a look at this talk as well https://www.ted.com/talks/joy_buolamwini_how_i_m_fighting_bias_in_algorithms.

```
Credit<-read.csv("Credit.csv")
str(Credit)
```

```
## 'data.frame':	400 obs. of  12 variables:
##  $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Income   : num  14.9 106 104.6 148.9 55.9 ...
##  $ Limit    : int  3606 6645 7075 9504 4897 8047 3388 7114 3300 6819 ...
##  $ Rating   : int  283 483 514 681 357 569 259 512 266 491 ...
##  $ Cards    : int  2 3 4 3 2 4 2 2 5 3 ...
##  $ Age      : int  34 82 71 36 68 77 37 87 66 41 ...
##  $ Education: int  11 15 11 11 16 10 12 9 13 19 ...
##  $ Gender   : Factor w/ 2 levels " Male","Female": 1 2 1 2 1 1 2 1 2 2 ...
##  $ Student  : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 2 ...
##  $ Married  : Factor w/ 2 levels "No","Yes": 2 2 1 1 2 1 1 1 1 2 ...
##  $ Ethnicity: Factor w/ 3 levels "African American",..: 3 2 2 2 3 3 1 2 3 1 ...
##  $ Balance  : int  333 903 580 964 331 1151 203 872 279 1350 ...
```

A scatter plot for each pair of the quantitative variables can be obtained using the command `pairs`

```
pairs(Balance~Age+Cards+Education+Income+Limit+Rating, Credit, col="steelblue3")
```

We observe the existence of linear dependency amongst some variables, such as `Rating` and `Limit`. These variables should not be included together in a linear model, as it would lead to numerical instabilities, and unreliable predictions. For example, considering the model consisting of the three variables `Age`, `Rating` and `Limit`, the value of the VIF for `Rating` and `Limit` indicates collinearity. One should drop one of the variables from the model, or combine them in some way.

```
library(car)
lm.fit = lm(Balance~Age+Rating+Limit, data=Credit)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Balance ~ Age + Rating + Limit, data = Credit)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -729.67 -135.82   -8.58  127.29  827.65
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -259.51752   55.88219  -4.644 4.66e-06 ***
## Age           -2.34575    0.66861  -3.508 0.000503 ***
## Rating         2.31046    0.93953   2.459 0.014352 *
## Limit          0.01901    0.06296   0.302 0.762830
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 229.1 on 396 degrees of freedom
## Multiple R-squared:  0.7536, Adjusted R-squared:  0.7517
## F-statistic: 403.7 on 3 and 396 DF,  p-value: < 2.2e-16
```

```r
vif(lm.fit)
```

```
##        Age     Rating      Limit
##   1.011385 160.668301 160.592880
```

Ridge Regression, the Lasso and relaetd techniques are implemented in the package `glmnet`

```r
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 1.9-8
```

We use the command `model.matrix` to automatically converts the qualitative variables into numerical variables. This is needed for the `glmnet` function which can only take numerical inputs. We remove the intercept from the model, see discussion pages 2-4 in lecture notes SL: RIDGE REGRESSION AND LASSO. Observe that `model.matrix` created two variables `EthnicityAsian` and `EthnicityCaucasian` for the predictor `Ethnicity`, containing originally three factors. The factor `African American` being the base factor by default. Because of ethical concerns, we drop these predictors from the model. For similar reasons, we drop as well the variable `Gender`.

```r
head(model.matrix(Balance~., Credit)) # First Column = Intercept
```

```
##   (Intercept) X  Income Limit Rating Cards Age Education GenderFemale
## 1           1 1  14.891  3606    283     2  34        11            0
## 2           1 2 106.025  6645    483     3  82        15            1
## 3           1 3 104.593  7075    514     4  71        11            0
## 4           1 4 148.924  9504    681     3  36        11            1
## 5           1 5  55.882  4897    357     2  68        16            0
## 6           1 6  80.180  8047    569     4  77        10            0
##   StudentYes MarriedYes EthnicityAsian EthnicityCaucasian
## 1          0          1              0                  1
## 2          1          1              1                  0
## 3          0          0              1                  0
## 4          0          0              1                  0
## 5          0          1              0                  1
## 6          0          0              0                  1
```

```r
train.X = model.matrix(Balance~., Credit)[,-c(1,2,9,12,13)]
train.Y = Credit$Balance
head(train.X)
```

```
##    Income Limit Rating Cards Age Education StudentYes MarriedYes
## 1  14.891  3606    283     2  34        11          0          1
## 2 106.025  6645    483     3  82        15          1          1
## 3 104.593  7075    514     4  71        11          0          0
## 4 148.924  9504    681     3  36        11          0          0
## 5  55.882  4897    357     2  68        16          0          1
## 6  80.180  8047    569     4  77        10          0          0
```

```r
head(train.Y)
```

```
## [1]  333  903  580  964  331 1151
```

We use the command `glmnet` to train the data using Ridge Regression and Lasso. The entry `alpha` indicates which method is used. The value `alpha=0` corresponds to ridge regression, and `alpha=1` to the lasso. In between values of `alpha` correspond to an elastic-net penalty. We estimate the parameters for a range of values of the tuning parameter λ, ranging from 10^5 to 10^{-2}.

```r
grid = 10^seq(5, -2, length=100)
ridge.fit = glmnet(train.X, train.Y, alpha=0, lambda=grid)
```

Note that by default, the function glmnet standardizes variables. Use `standardize=FALSE` if you do not want glmnet to standardize.

```r
args(glmnet)
```

```
## function (x, y, family = c("gaussian", "binomial", "poisson",
##     "multinomial", "cox", "mgaussian"), weights, offset = NULL,
##     alpha = 1, nlambda = 100, lambda.min.ratio = ifelse(nobs <
##         nvars, 0.01, 1e-04), lambda = NULL, standardize = TRUE,
##     intercept = TRUE, thresh = 1e-07, dfmax = nvars + 1, pmax = min(dfmax *
##         2 + 20, nvars), exclude, penalty.factor = rep(1, nvars),
##     lower.limits = -Inf, upper.limits = Inf, maxit = 1e+05, type.gaussian = ifelse(nvars <
##         500, "covariance", "naive"), type.logistic = c("Newton",
##         "modified.Newton"), standardize.response = FALSE, type.multinomial = c("ungrouped",
##         "grouped"))
## NULL
```

For each value of λ, glmnet returns a vector of estimated coefficients, which can be accessed using `coef()`.

```r
dim(coef(ridge.fit))
```

```
## [1]   9 100
```

Expect coefficients to be close the the least squares estimates for small values of λ. In addition, as λ increases, the size of the coefficients are shrinking. Compare the following estimates:

```r
grid[20]  # Value of lambda
```

```
## [1] 4534.879
```

```r
coef(ridge.fit)[,20]
```

```
##  (Intercept)       Income        Limit       Rating        Cards 
## 352.62843891   0.42219591   0.01402781   0.20973004   2.56585994 
##          Age    Education   StudentYes   MarriedYes 
##  -0.04388717  -0.06297835  36.45865958  -0.84991432
```

```r
sqrt(sum(coef(ridge.fit)[-1,20]^2)) # Sum of the squared estimated coefficient
```

```
## [1] 36.56184
```

```r
grid[100] # Value of lambda
```

```
## [1] 0.01
```

```r
coef(ridge.fit)[,100]
```

```
##  (Intercept)       Income        Limit       Rating        Cards 
## -478.8370871   -7.7901130    0.1785274    1.3168139   16.9765869 
##          Age    Education   StudentYes   MarriedYes 
##   -0.6413354   -1.0399935  424.7658276   -7.6857956
```

```r
sqrt(sum(coef(ridge.fit)[-1,100]^2)) # Sum of the squared estimated coefficient
```

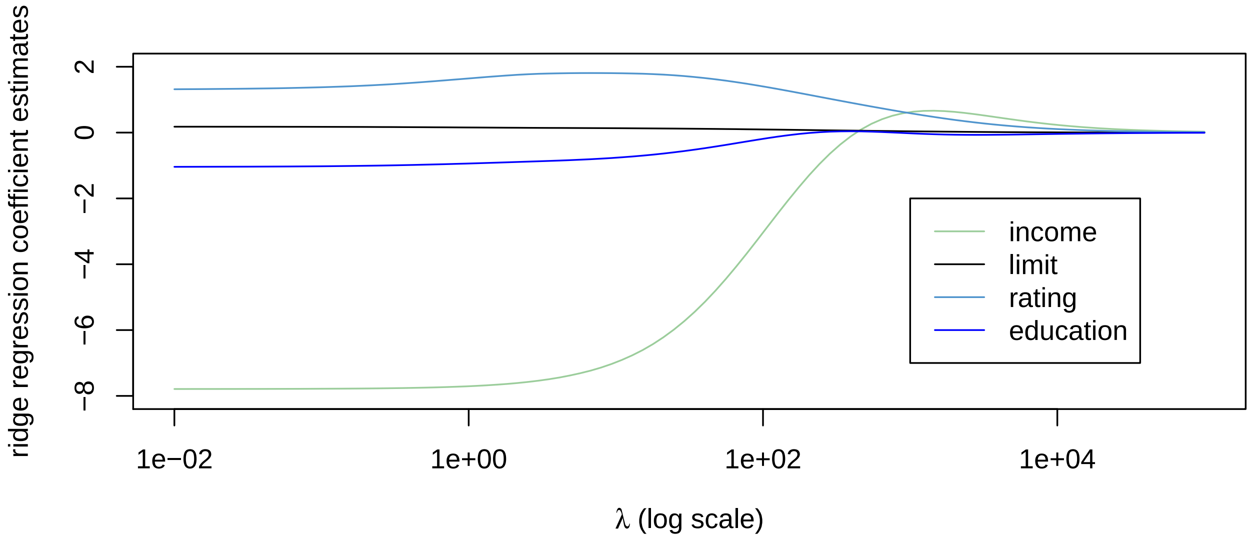
```
## [1] 425.2496
```

```r
lm.fit = lm(train.Y~train.X)
coef(lm.fit)  # LS estimate
```

```
##       (Intercept)     train.XIncome      train.XLimit     train.XRating 
##      -473.6511986        -7.7933184         0.1930547         1.1007640 
##      train.XCards        train.XAge  train.XEducation train.XStudentYes 
##        18.0206890        -0.6385828        -1.0957774       425.4586749 
## train.XMarriedYes 
##        -7.2149398
```

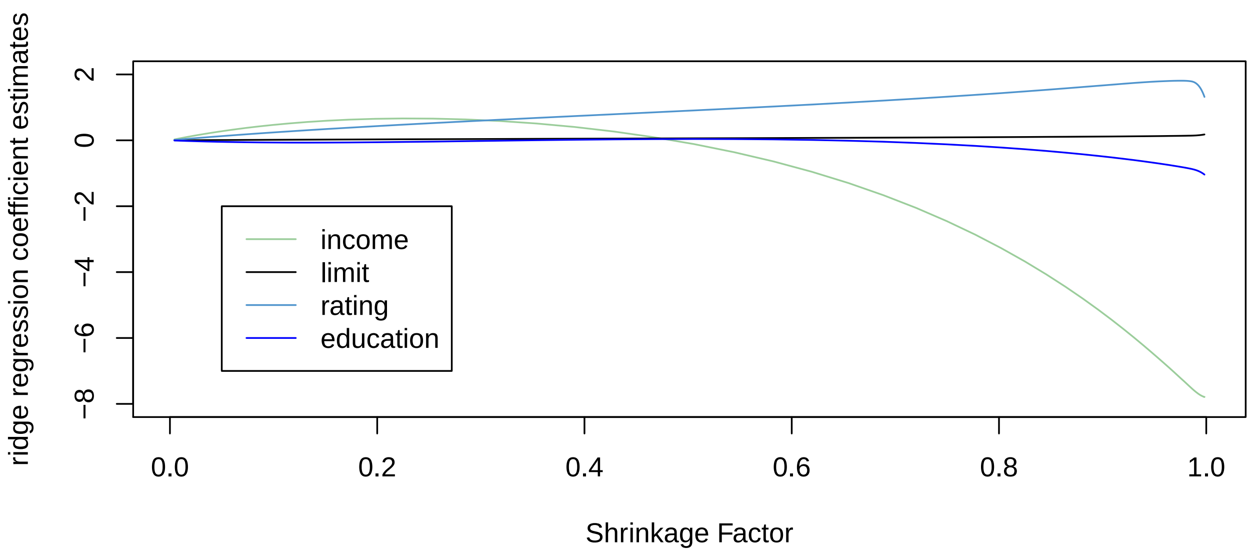
We plot the evolution of the value of the coefficients as a function of λ.

```r
plot(grid, coef(ridge.fit)[2,], ylim = c(-8, 2), log="x", type="l",col="darkseagreen3",
     xlab= expression(paste(lambda, " (log scale)")),
     ylab="ridge regression coefficient estimates")           # variable 'income'
lines(grid, coef(ridge.fit)[3,], type="l", col="black")       # variable 'limit'
lines(grid, coef(ridge.fit)[4,], type="l", col="steelblue3")  # variable 'rating'
lines(grid, coef(ridge.fit)[7,], type="l", col="blue")        # variable 'education'
legend(1000, -2, c("income","limit","rating","education"), lty=c(1,1,1,1),
       col=c("darkseagreen3","black","steelblue3","blue"))
```



Alternatively, as λ increases, the $||.||_2$ norm of the vector of ridge estimates $\hat{\beta}_\lambda$ decreases. We can plot the value of the coefficients as a function of $||\hat{\beta}_\lambda||_2^2/||\hat{\beta}||_2^2$, where $\hat{\beta}$ denotes the LS estimates.

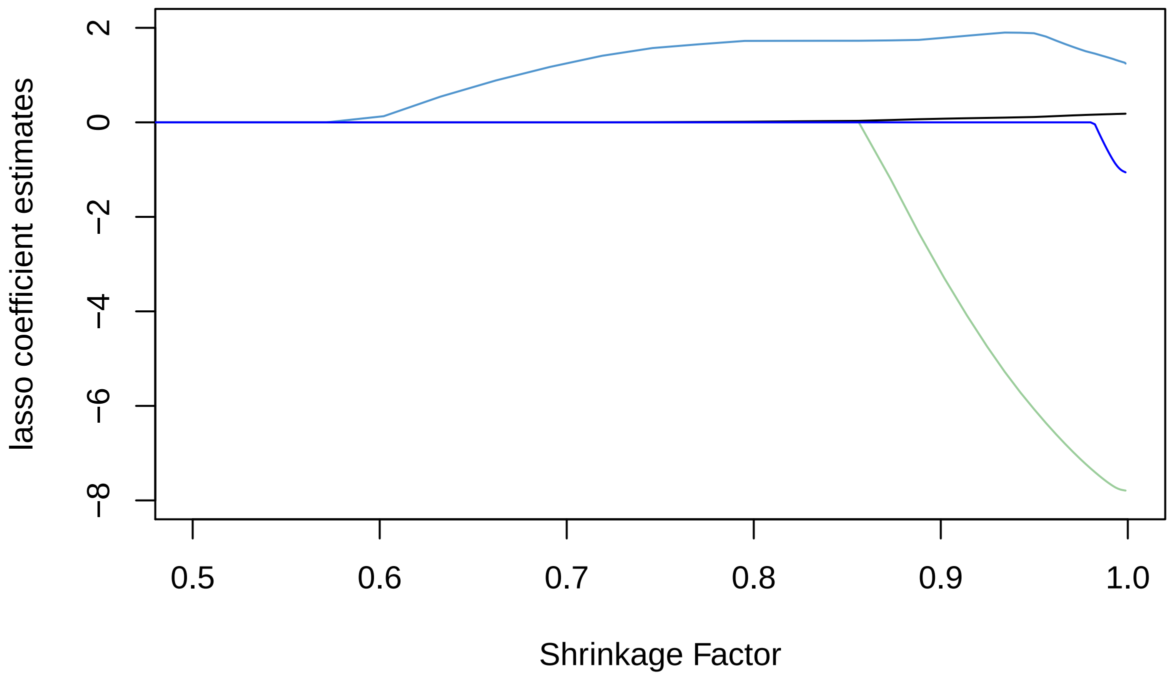
```
ratiol2 = sqrt(colSums(coef(ridge.fit)[-1,]^2)/sum(coef(lm.fit)[-1]^2))
plot(ratiol2, coef(ridge.fit)[2,], ylim = c(-8, 2), type="l", col="darkseagreen3",
     xlab="Shrinkage Factor", ylab="ridge regression coefficient estimates")
lines(ratiol2, coef(ridge.fit)[3,], type="l", col="black")
lines(ratiol2, coef(ridge.fit)[4,], type="l", col="steelblue3")
lines(ratiol2, coef(ridge.fit)[7,], type="l", col="blue")
legend(0.05, -2, c("income","limit","rating","education"), lty=c(1,1,1,1),
       col=c("darkseagreen3","black","steelblue3","blue"))
```



We repeat the analysis for the lasso, setting `alpha=1` in the command `glmnet`. We plot a subset of the coefficient estimates as a function of $||\hat{\beta}_\lambda||_1/||\hat{\beta}||_1$, where $\hat{\beta}_\lambda$ denotes the lasso estimate, and $||.||_1$ the ℓ_1-norm.

```
grid = 10^seq(5, -3, length=100)
lasso.fit = glmnet(train.X, train.Y, alpha=1, lambda=grid)
ratiol1=sqrt(colSums(abs(coef(ridge.fit)[-1,]))/sum(abs(coef(lm.fit)[-1])))
plot(ratiol1, coef(lasso.fit)[2,], ylim = c(-8, 2), type="l", col="darkseagreen3",
```

```r
     xlim=c(.5,1), xlab="Shrinkage Factor", ylab="lasso coefficient estimates")
lines(ratio11, coef(lasso.fit)[3,], type="l", col="black")
lines(ratio11, coef(lasso.fit)[4,], type="l", col="steelblue3")
lines(ratio11, coef(lasso.fit)[7,], type="l", col="blue")
legend(0.1,-4,c("income","limit","rating","cards"), lty=c(1,1,1,1),
       col=c("red","black","magenta","blue"))
```



Finally, the coefficients returned by `glmnet` differ depending on how the input variables are scaled. Consider the following twho scenarios: (a) use `glmnet` with manually standardised variables, using `standardize=FALSE`, and (b) non manually standardised input variables, and `standardize=TRUE` in `glmnet`.

```r
# First, use glmnet with standardized inputs
m.X <- colMeans(train.X); n=dim(Credit)[1]
sd.X <- sqrt((1-1/n)*apply(train.X, 2, var)) # Renormalise by n instead of (n-1)
m.X <- matrix(rep(m.X, dim(train.X)[1]), nrow=dim(train.X)[1],
              ncol=dim(train.X)[2], byrow=TRUE)
sd.X <- matrix(rep(sd.X, dim(train.X)[1]), nrow=dim(train.X)[1],
               ncol=dim(train.X)[2], byrow=TRUE)

# Case (a)
ridge.fit1 = glmnet((train.X-m.X)/sd.X, train.Y, alpha=0, lambda=25, standardize=FALSE)
coef(ridge.fit1)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  520.015000
## Income      -210.278163
## Limit        281.130123
## Rating       268.905257
## Cards         21.130488
## Age          -15.295031
```

```
## Education     -1.867084
## StudentYes   118.674146
## MarriedYes    -5.156212
```

```r
# Case (b)
ridge.fit2 = glmnet(train.X, train.Y, alpha=0, lambda=25)
coef(ridge.fit2)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) -426.2437625
## Income        -5.9737792
## Limit          0.1219489
## Rating         1.7401423
## Cards         15.4286724
## Age           -0.8877888
## Education     -0.5981754
## StudentYes   395.5804858
## MarriedYes   -10.5838063
```

You should observe that all coefficient estimates differ in cases (a) and (b). This may seem worrying at first, as in (a) we manually standardize coefficients, while in (b) we rely on `glmnet` to do it automatically. This raises the question of the interpretability of the coefficient estimates returned by `glmnet`. First, note that the intercept value in (a) corresponds to the mean value of the response variable.

```r
mean(train.Y)
```

```
## [1] 520.015
```

This observation is in agreement with the discussion on page 4 of the lecture notes. It turns out that the estimate of the intercept of the `glmnet` function is returned on the *original* scale, that is the scale on which you input the training data to the `glmnet` command. Compare now the coefficient estimate of (a) with those of (b) properly rescaled, corresponding respectively to $\hat{\beta}_j^s$ and $\hat{\beta}_j$ in the notation p.4 of the lecture notes: coefficients $\hat{\beta}_j^s$ correspond to `ridge.fit1` while coefficients $\hat{\beta}_j$ correspond to `ridge.fit2`, and satisfy $\hat{\beta}_0 = \hat{\beta}_0^s - \sum_{j=1}^d \hat{\beta}_j \bar{x}_j$, and $\hat{\beta}_j = \hat{\beta}_j^s / \hat{\sigma}_j$, for $j = 1, ..., d$.

```r
m.X <- colMeans(train.X); sd.X <- sqrt((1-1/n)*apply(train.X, 2, var))
coef(ridge.fit2)[1]
```

```
## [1] -426.2438
```

```r
coef(ridge.fit1)[1] - sum(m.X*coef(ridge.fit2)[2:9])
```

```
## [1] -426.2438
```

```r
coef(ridge.fit2)[2:9]
```

```
## [1]  -5.9737792   0.1219489   1.7401423  15.4286724  -0.8877888  -0.5981754
## [7] 395.5804858 -10.5838063
```

```r
coef(ridge.fit1)[2:9]/sd.X
```

```
##      Income       Limit      Rating       Cards         Age   Education 
##  -5.9737792   0.1219489   1.7401423  15.4286724  -0.8877888  -0.5981754 
##  StudentYes  MarriedYes 
## 395.5804858 -10.5838063
```