Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 201 (2017) 746–755

Procedia Engineering

www.elsevier.com/locate/procedia

3rd International Conference "Information Technology and Nanotechnology, ITNT-2017, 25-27 April 2017, Samara, Russia

# Comparing Ridge and LASSO estimators for data analysis

L.E. Melkumova[a,*], S.Ya. Shatskikh[b]

[a]Mercury Development Russia, 110k1, Avrory street, Samara, 443069, Russia
[b]Samara National Research University, 34, Moskovskoe shosse, Samara, 443086, Russia

**Abstract**

This paper is devoted to the comparison of Ridge and LASSO estimators. Test data is used to analyze advantages of each of the two regression analysis methods. All the required calculations are performed using the R software for statistical computing.


Peer-review under responsibility of the scientific committee of the 3rd International Conference "Information Technology and Nanotechnology".

*Keywords:* linear regression; Ridge regression; LASSO; cross-validation

## 1. Multivariate linear regression. Ordinary least squares

Linear regression considers the following relationship between some number of explanatory variables $x_1, \ldots, x_{k-1}$ (predictors) and a dependent variable $y$ (response).

$$y = b_0 + b_1 x_1 + \ldots + b_{k-1} x_{k-1} + \varepsilon, \tag{1}$$

where $\varepsilon$ is interpreted as a random observational error or fluctuations.

The measurements are conducted $n$ times so that one has $n$ values of $y$ for $n$ sets of $x_j$.

$$y_i = b_0 + b_1 x_{i1} + \ldots + b_{k-1} x_{i\,k-1} + \varepsilon_i, \quad i = \overline{1, n}, \tag{2}$$

where $x_{ij}$ is the $i$th observation of $x_j$. The $\varepsilon_i$ are not observed directly.

The equations (2) can be expressed in the matrix form after adding the parameters

$$x_{10} = x_{20} = \ldots = x_{n0} = 1. \tag{3}$$

Then

$$\mathbf{Y} = \mathbf{XB} + \mathcal{E}, \tag{4}$$

where

$$\mathbf{Y} = [y_i]_n, \quad \mathbf{X} = \left[x_{ij}\right]_{n\times k}, \quad \mathbf{B} = \left[b_j\right]_k, \quad \mathcal{E} = [\varepsilon_i]_n. \tag{5}$$

* Corresponding author.
*E-mail address:* lana.melkumova@gmail.com

1877-7058 
Peer-review under responsibility of the scientific committee of the 3rd International Conference "Information Technology and Nanotechnology".
10.1016/j.proeng.2017.09.615

The coordinates $b_0, b_1, \ldots, b_{k-1}$ of the vector $\mathbf{B}$ are unknown. The goal of the regression analysis is to estimate the vector $\mathbf{B}$ based on the multivariate observations.

$$[\mathbf{X}, \mathbf{Y}] = \begin{bmatrix} x_{10} & x_{11} & \ldots & x_{1\,k-1} & y_1 \\ x_{20} & x_{21} & \ldots & x_{2\,k-1} & y_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{n0} & x_{n1} & \ldots & x_{n\,k-1} & y_n \end{bmatrix}. \tag{6}$$

A traditional approach to this problem is to use the ordinary least squares (OLS) estimator where

$$\sum_{i=1}^{n} \left( y_i - \sum_{j=0}^{k-1} b_j \, x_{ij} \right)^2 \mapsto \min. \tag{7}$$

The OLS estimates of the unknown coefficients $b_0, b_1, \ldots, b_{k-1}$ minimize (7):

$$\widehat{\mathbf{B}} = [\widehat{b}_j]_k. \tag{8}$$

Given

$$\det \mathbf{X}^t\mathbf{X} > 0 \tag{9}$$

the OLS estimates can be calculated using the following formula

$$\widehat{\mathbf{B}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y}. \tag{10}$$

Let us denote

$$\widehat{\mathbf{Y}} := \mathbf{X}\widehat{\mathbf{B}}. \tag{11}$$

One can rewrite this equation in the coordinate form as follows

$$\widehat{y}_i := \widehat{b}_0 + \widehat{b}_1 x_{i1} + \ldots + \widehat{b}_{k-1} x_{i\,k-1}, \quad i = \overline{1,n}. \tag{12}$$

Here $\widehat{y}$ is the predicted response value that corresponds to the predictor values $x_1, \ldots, x_{k-1}$.
The residual sum of squares (RSS) measures the discrepancy between the data and the estimation model.

$$RSS := \sum_{i=1}^{n} (\widehat{y}_i - y_i)^2. \tag{13}$$

The coefficient of determination

$$R^2 := 1 - \frac{\sum\limits_{i=1}^{n} (\widehat{y}_i - y_i)^2}{\sum\limits_{i=1}^{n} (y_i - \overline{y})^2} \in [0,\, 1] \tag{14}$$

also measures the quality of the regression model: the closer it is to 1, the better the regression model (11) fits the data (6).

### *1.1. Data standardization*

Data standardization is often used in the linear regression analysis (see [1]). Namely, by denoting

$$\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}, \quad \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad S_y^2 = \sum_{i=1}^{n} (y_i - \overline{y})^2, \quad S_j^2 = \sum_{i=1}^{n} \left( x_{ij} - \bar{x}_j \right)^2, \quad j = \overline{1, k-1}, \tag{15}$$

we get centered and normalized variables for the initial sample (6)

$$v_i := \frac{y_i - \overline{y}}{S_y}, \qquad w_{ij} := \frac{x_{ij} - \bar{x}_j}{S_j}, \quad i = \overline{1,n}, \quad j = \overline{1, k-1}. \tag{16}$$

If we denote

$$\mathbf{V} = [v_i]_n\,, \qquad \mathbf{W} = \left[w_{ij}\right]_{n\times(k-1)}\,, \tag{17}$$

in case if $\det \mathbf{W}^t\mathbf{W} > 0$, the OLS estimates for the standardized model can be calculated using the formula

$$\widehat{B} = (\mathbf{W}^t\mathbf{W})^{-1}\mathbf{W}^t\mathbf{V}. \tag{18}$$

There are several advantages of using standardized data for linear regression. Firstly, with standardized data the solution does not depend on the measurement scale. The predictors $x_j$ may be measured in different scales while standardized predictors $w_j$ are reduced to the same "neutral" scale. Secondly, the predictor input is more likely to depend on the relative value $w_j$ rather than the absolute value $x_j$.

## 2. Ridge regression and the LASSO

It is often the case that the matrix $\mathbf{X}^t\mathbf{X}$ is "close" to singular. This phenomenon is called *multicollinearity* in a multiple regression model. In this case we still can obtain the OLS estimates, but they will likely have "bad" statistical properties. Slight variations in the statistical data (like adding or removing a few observations) will lead to significant changes in the coefficient estimates.

One of the ways to detect multicollinearity in the regression data is to use the *variance inflation factors* $VIF_j$, $j = \overline{1,k}$ (see [2]). In practice, when $VIF_j > 5$ (or $VIF_j > 10$) for at least one $j$, it indicates a high multicollinearity for the regression matrix $\mathbf{X}$.

The two regression analysis methods Ridge and LASSO perform regularization of the estimated coefficients and can sometimes overcome the disadvantages of the OLS estimator.

*Prediction accuracy.* If the relationship between the variables is almost linear and the number of predictors $k$ is much less than the sample size ($k \ll n$), then it is likely that the plain OLS estimator would yield good results. However, if $k$ is not much less than $n$, then the OLS estimates will tend to have high variance and low accuracy. If $k > n$, the OLS procedure will not give a unique solution and the variance of the estimates will become infinite. The regularization techniques often allow reducing the estimate variance at the cost of introducing a slight bias. As a result, the prediction accuracy increases.

*Model interpretability.* It is often the case that the regression model contains a big number $k$ of predictors while many of them do not really affect the response value. If we exclude these variables from the model, it would be easier to interpret. In this sense, the Ridge and LASSO estimators provide an alternative to "subset selection" techniques that allow reducing the number of predictors in the regression model. They produce linear models where the coefficient estimates are close to zero (or equal zero).

## 3. Ridge regression

*The Ridge estimate* of an unknown vector $B$ for standardized observations $\{\mathbf{W}, \mathbf{V}\}$ is given by

$$\widetilde{B_\lambda} := \left(\mathbf{W}^t\mathbf{W} + \lambda\mathbb{I}\right)^{-1}\mathbf{W}^t\mathbf{V}, \tag{19}$$

where $\mathbb{I}$ is the identity matrix. $\lambda > 0$ is called the *regularization parameter*.

We will denote the coordinate form of the Ridge estimate by

$$\widetilde{B}_\lambda \;=\; \left[\tilde{\beta}_j(\lambda)\right]_{k-1}\,. \tag{20}$$

By adding the "ridge" parameter $\lambda$ to the diagonal elements of the matrix $\mathbf{W}^t\mathbf{W}$, we can transform the ill-conditioned matrix $\mathbf{W}^t\mathbf{W}$ into the well-conditioned matrix $(\mathbf{W}^t\mathbf{W} + \lambda\mathbb{I})$. This way we avoid usual problems with the ill-conditioned matrix inversion. However, it is worth noting that, unlike the OLS, the Ridge estimate $\widetilde{B}_\lambda$ is biased.

It can be shown that the Ridge estimate $\widetilde{B}_\lambda$ is the solution of the following equivalent minimization problems:

$$\begin{aligned}
&1^\circ \;\; \| \mathbf{V} - \mathbf{W}B \|^2 + \lambda \| B \|^2 \longmapsto \min,\\
&2^\circ \text{ For all } \lambda > 0, \text{ there is a } t(\lambda) > 0 \text{ such that}\\
&\| \mathbf{V} - \mathbf{W}B \|_2 \longmapsto \min \text{ subject to } \| B \|_2 \le t(\lambda).
\end{aligned} \tag{21}$$

So the Ridge estimate can be viewed as an OLS estimate with an additional penalty imposed on the coefficient vector.

## 4. LASSO

The LASSO[1] estimate $\widetilde{B}_\lambda$ (see [3], [4]) is the solution of the following equivalent minimization problems for standardized observations $\{\mathbf{W}, \mathbf{V}\}$.

$$\begin{aligned} &1^\circ \ \| \mathbf{V} - \mathbf{W}B \|^2 + \lambda \| B \|_1 \longmapsto \min, \\ &2^\circ \text{ For all } \lambda > 0, \text{ there is a } t(\lambda) > 0 \text{ such that} \\ &\| \mathbf{V} - \mathbf{W}B \|^2 \longmapsto \min \text{ subject to } \| B \|_1 \le t(\lambda), \end{aligned} \tag{22}$$

where $\|B\|_1 = \sum_{j=1}^{k-1} |\beta_j|$.

The penalty on the coefficient vector $\beta_j$, $j = \overline{1, k-1}$ imposed by LASSO is slightly different from Ridge. In case of LASSO, the $\lambda$ parameter is multiplied by the $\ell_1$-norm of the vector $(\beta_1, \ldots, \beta_{k-1})$ while Ridge uses the $\ell_2$-norm.

One of the positive effects of this change in terms of model interpretability is the fact that the LASSO, unlike Ridge regression, results in a model where some coefficient estimates are exactly equal to zero when $\lambda$ is large. In other words, the LASSO regularization additionally performs variable selection which makes the model easier to interpret.

As in the case of Ridge, different $\lambda$ values produce different $\widetilde{B}_\lambda$ vectors. That is why it is important to select a proper $\lambda$ value. Section 5 describes the *cross-validation* technique that can be used for this purpose.

## 5. Selecting the $\lambda$ value for Ridge and LASSO. Cross-validation

Cross-validation is the technique that can be used to find a "proper" value for the $\lambda$ parameter given a sample (see [3], [4], [5]). By "proper" here we mean that we are trying to find a $\lambda$ that would allow to predict the response values with the highest accuracy. It is clear that the $\lambda$ values that are too small can lead to ovefitting when the model would tend to describe the noise in the data. Too large $\lambda$ values, on the contrary, would lead to underfitting when the procedure cannot capture the underlying relationship. In both cases we will get a high error value when calculated on the test data (a set of observations not included into the initial sample).

To perform cross-validation, we first divide the initial data into two subsets: one is called the *train set* and the other one is called the *test set*. Then the train set is used to calculate the coefficient estimates. These estimates are then validated on the test set.

Let us now describe the algorithm in some more detail. First the initial data set is randomly divided into $Q$ blocks of equal length. One of the blocks is assigned the role of the test set while the remaining $Q - 1$ blocks together constitute the train set. In practice the number of blocks $Q$ is usually selected to be 5 or 10. Next we choose a grid of values $\lambda = [\lambda_s]$ and calculate the regression coefficients for each $\lambda_s$ value. Given these regression coefficients, we then compute the residual sum of squares.

$$RSS^q_{\lambda_s} = \sum_{i=1}^{n} \left( y_i - \sum_{j=0}^{k-1} \widehat{b_j}(q, \lambda_s)\, x_{ij} \right)^2, \tag{23}$$

where $q = \overline{1, Q}$ is the index of the block selected as the test set. One can obtain the average of these RSS values over all blocks.

$$MSE_{\lambda_s} = \frac{1}{Q} \sum_{q=1}^{Q} RSS^q_{\lambda_s}. \tag{24}$$

[1] Least Absolute Shrinkage and Selection Operator.

$\lambda$ is then set equal to $\lambda_s$ that gives the minimum $MSE_{\lambda_s}$.

Another popular approach utilizes the "one-standard-error rule" (see for example [6]). For each $MSE_{\lambda_s}$ the standard error of the mean is calculated. Then we select the largest $\lambda_s$ for which the $MSE_{\lambda_s}$ is within one standard error of the minimum $MSE$ value. This way we obtain a "more regularized" model while increasing the $MSE$ by not more than one standard error.

## 6. Ridge and LASSO applied to test data

Let us now apply the OLS, Ridge and LASSO regression techniques to real-world data. We will be using observations of different wine parameters that can be found on the UCI Machine Learning Repository ([7]). The Wine Quality data set ([8], [9]) describes red and white variants of the "Vinho Verde" wine from the Minho province in the north of Portugal. In this work, we only analyze the red wine data. The predictors are various physicochemical parameters such as density, acidity, the amount of sugar and alcohol. The response is the wine quality score between 0 and 10.

We are going to build regression models that describe how the wine quality score depends on the physicochemical characteristics of the wine using OLS, Ridge and LASSO. We will be using the R software for all calculations, namely its `glmnet` package that allow fitting Ridge and LASSO linear models.

First we need to load the `glmnet` library into R.

```
library(glmnet)
```

The wine data is in CSV format. One can use the `read.csv2` command for loading the file into R. The data is saved into the `wine` variable for future use.

```
wine = read.csv2("winequality-red.csv", na.strings="N/A", dec=".")
```

Each observation includes 12 wine parameters. 11 of them are physical and chemical characteristics that we consider as predictors. The 12th is the wine quality score, an integer value between 0 and 10 that we consider as the response. To list column names of the `wine` matrix, the `names` command can be used (see table 1).

```
names(wine)
```

Table 1. The list of predictors (column names) of the `wine` matrix.

| | | | | | |
|---|---|---|---|---|---|
| 1. fixed.acidity | 2. volatile.acidity | 3. citric.acid | 4. residual.sugar | 5. chlorides | 6. free.sulfur.dioxide |
| 7. total.sulfur.dioxide | 8. density | 9. pH | 10. sulphates | 11. alcohol | 12. quality |

The data matrix for the red wine contains 1599 observations. You can check this using the `dim` command for getting matrix dimensions.

```
dim(wine)
[1] 1599  12
```

To be able to measure quality of the regression model, we are going to verify results on a test set. We randomly divide the whole sample into two subsets: the train set and the test set. Here the `set.seed` command with a fixed input value of 1 is used to get results that can later be reproduced by the reader.

```
wine.pred.names = names(wine)[1 : length(names(wine)) -1]
wine.pred = wine[, wine.pred.names]

set.seed(1)
train=sample(1:nrow(wine.pred), nrow(wine.pred)/2)
test=(-train)
```

```
x = model.matrix(~., wine.pred[train, ])[, -1]
y = wine$quality[train]
x.test = model.matrix(~., wine.pred[test, ])[, -1]
y.test = wine$quality[test]
```

We will store the predictor matrix in the `x` variable and the response vector in the `y` variable. For the test set, we will be using variable names `x.test` and `y.test`.

To perform the OLS regression, the `lm` command can be used. The OLS coefficient estimates for the wine data are given in the first row of table 3.

```
lm.mod = lm(y~., data = data.frame(x))
```

The `summary` command gives the overview of the different regression model parameters. One of them is the coefficient of determination $R^2$ that in our case equals 0.3334006.

```
summary(lm.mod)$r.squared
[1] 0.3334006
```

To check the data for multicollinearity, we calculate the variance inflation factors $VIF_j$. One can do this using the `vif` command from the `car` package.

```
library(car)
vif(lm.mod)
```

Table 2. Variance inflation factors $VIF_j$, $j = \overline{1,11}$ for the red wine data.

| $VIF_1$ | $VIF_2$ | $VIF_3$ | $VIF_4$ | $VIF_5$ | $VIF_6$ | $VIF_7$ | $VIF_8$ | $VIF_9$ | $VIF_{10}$ | $VIF_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 7.175034 | 1.777550 | 3.424212 | 1.673862 | 1.513932 | 2.043192 | 2.235502 | 5.874435 | 3.239267 | 1.447768 | 2.852950 |

The $VIF_j$ values are given in the table 2. Some of these values are rather high, like $VIF_1$ and $VIF_8$. This indicates a considerable though not critical multicollinearity: all the $VIF_j < 10$.

Let us now move to the Ridge regression. As it was already mentioned, different $\lambda$ values will produce different coefficient estimates $\widehat{b_i}$. In R, we can visualize how the $\widehat{b_i}$ estimates change depending on $\lambda$ for a given sample.

To do this, we first choose a range of $\lambda$ values and build regression models for each $\lambda$ value from this range using a handy `glmnet` function from the `glmnet` package. Note that the function standardizes input data - both predictors and response are normalized and centered before performing the regression. Still, the result is always given in the original data scale. For this reason in particular, we get the non-zero intercept value $\widehat{b_0}$.

```
lambda.grid = 10^seq(5, -2, length=100)
ridge.mod = glmnet(x, y, alpha=0, lambda=lambda.grid)
```

In this case $\lambda$ takes 100 different values from $10^{-2}$ to $10^5$ and the power increases with a constant step $\frac{7}{99}$. The following command can be used to plot the coefficient estimates against $\lambda$.

```
plot(ridge.mod, xvar='lambda')
```

The command generates the plot given in the figure (1) on the left. One can see that, as $\lambda$ increases, the coefficient estimates $\widehat{b_i}$ are "shrunk" towards zero which means that the norm of the estimates vector $\left\|\widehat{b}(\lambda)\right\|_2$ decreases. At the same time, individual coefficients can increase on specific intervals (see for example the red line that denotes the *density* coefficient estimate).

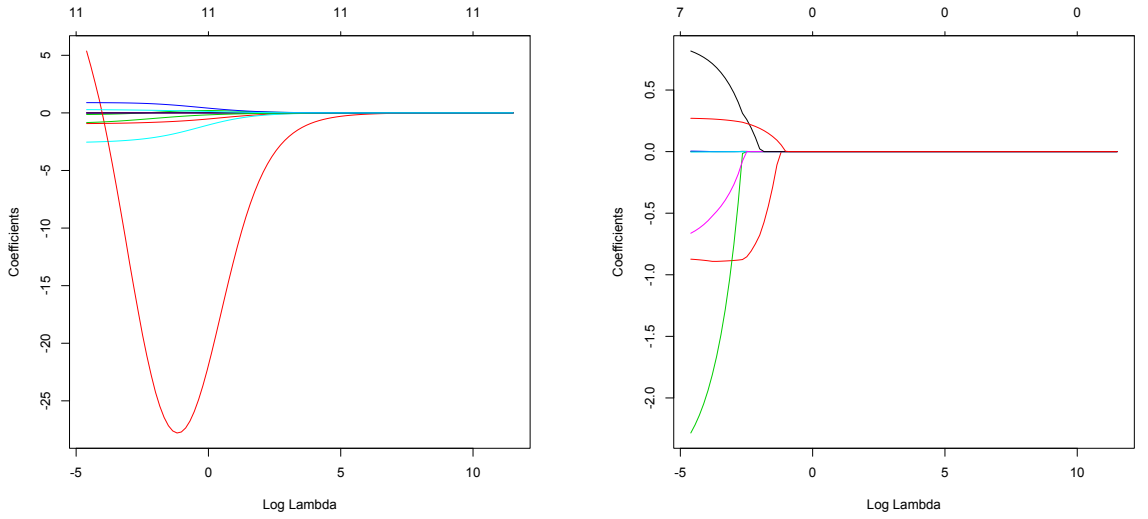


Fig. 1. Coefficient estimates $\beta_i$ for Ridge regression (on the left) and the LASSO (on the right) for the red wine data plotted versus $\log \lambda$. The upper part of the plot shows the number of non-zero coefficients $\widehat{b_i}$ in the regression model for a given $\log \lambda$.

The upper part of the plot shows the number of non-zero coefficient estimates $\widehat{b_i}$ for a given value of $\log \lambda$. As can be seen, for Ridge this number is constant for all the $\lambda$ values and equals the number of predictors in the data. Thus, although the Ridge regression shrinks the coefficient estimates close to zero, even large $\lambda$ values do not produce estimates exactly equal to zero.

It is worth comparing these coefficient estimates with those generated by LASSO for the same $\lambda$ values. We will again use the `glmnet` to perform LASSO regression for the same initial data and the same `lambda.grid` vector. The only thing that needs to be changed is the `alpha` parameter that in this case takes the value of 1.

```
lasso.mod = glmnet(x, y, alpha=1, lambda=lambda.grid)
plot(lasso.mod, xvar='lambda')
```

The output of the `plot` command is given in the figure (1) on the right. It can be seen that the LASSO regression also tends to "shrink" the regression coefficients to zero as $\lambda$ increases. The reader can tell this by looking at the numbers in the upper part of the plot which again mean the number of non-zero coefficients in the regression model. For instance, when $\log \lambda = -5$ there are 7 non-zero coefficients and when $\log \lambda = 0$ we get the model where all the coefficients are zero. That is, the LASSO regression performs variable selection when $\lambda$ is large enough.

Now let us select the $\lambda$ parameter using the cross-validation procedure. We utilize the `cv.glmnet` command from the `glmnet` package. Again, to achieve reproducible results we use the `set.seed` function. The `cv.glmnet` commands for Ridge and LASSO differ only by the `alpha` parameter value.

```
set.seed(1)
ridge.cv.out = cv.glmnet(x, y, alpha=0)
set.seed(1)
lasso.cv.out = cv.glmnet(x, y, alpha=1)
```

The cross-validation results can be visualized with the `plot` command (see figure (2)).

```
plot(ridge.cv.out)
plot(lasso.cv.out)
```

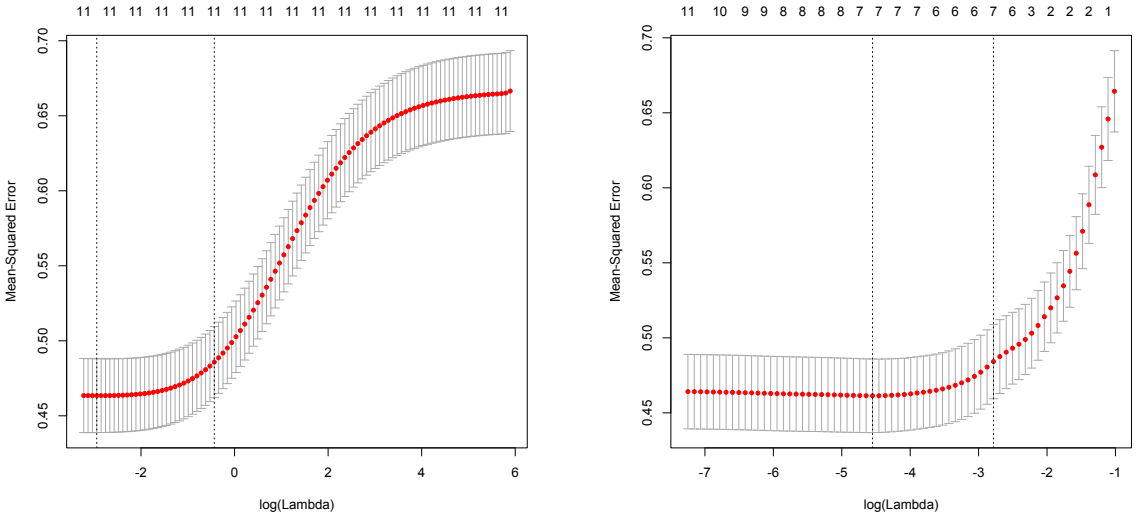


Fig. 2. Cross-validated estimate of the mean squared prediction error for Ridge (on the left) and LASSO (on the right), as a function of $\log \lambda$. The upper part of the plot shows the number of non-zero coefficients $\widehat{b_i}$ in the regression model for a given $\log \lambda$. The dashed lines show the location of the function minimum and the "one-standard-error" location.

The plots depict the mean squared prediction error $MSE_\lambda$ against $\log \lambda$. The gray bars at each point show $MSE_\lambda$ plus and minus one standard error. One of the vertical dashed lines shows the location of the minimum of $MSE$. The second dashed line shows the point selected by the "one-standard-error" rule.

Using the cross-validation results, we now get proper values for $\lambda$.

```
ridge.bestlam = ridge.cv.out$lambda.min
ridge.lam1se = ridge.cv.out$lambda.1se
lasso.bestlam = lasso.cv.out$lambda.min
lasso.lam1se = lasso.cv.out$lambda.1se
```

In our case Ridge regression gives $ridge.bestlam = 0.05257397$ and $ridge.lam1se = 0.6481565$. The LASSO yields $lasso.bestlam = 0.01056334$ and $lasso.lam1se = 0.06186968$ correspondingly.

The coefficient estimates computed for each of these $\lambda$ values are given in the table 3. To calculate the estimates, one needs to perform regression for each $\lambda$. The coefficients are then printed using the `coef` function.

```
ridge.mod.best = glmnet(x, y, alpha=0, lambda=ridge.bestlam)
coef(ridge.mod.best)
ridge.mod.1se = glmnet(x, y, alpha=0, lambda=ridge.lam1se)
coef(ridge.mod.1se)

lasso.mod.best = glmnet(x, y, alpha=1, lambda=lasso.bestlam)
coef(lasso.mod.best)
lasso.mod.1se = glmnet(x, y, alpha=1, lambda=lasso.lam1se)
coef(lasso.mod.1se)
```

As can be seen, both Ridge and LASSO produce a more "regularized" model when moving from *bestlam* to *lam1se* in the sense that the coefficient estimates get more "shrunk" towards zero. In case of LASSO, the number of non-zero coefficients did not change although their indexes did. The $b_1$ value, which was zero at $\lambda = lasso.bestlam = 0.01056334$, is non-zero again for $\lambda = lasso.lam1se = 0.05980285$ while $b_{10}$ becomes equal to zero.

Let us now compare the $RSS$ values for all the models. We will first calculate the $RSS$ on the train set and then move to the test set.

Table 3. Coefficient estimates for the red wine data for different regression techniques.

| Method | $\mathbf{b_0}$ | $\mathbf{b_1}$ | $\mathbf{b_2}$ | $\mathbf{b_3}$ | $\mathbf{b_4}$ | $\mathbf{b_5}$ | $\mathbf{b_6}$ | $\mathbf{b_7}$ | $\mathbf{b_8}$ | $\mathbf{b_9}$ | $\mathbf{b_{10}}$ | $\mathbf{b_{11}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OLS | -8.915 | -0.013 | -0.942 | -0.164 | -0.005 | -2.599 | 0.007 | -0.004 | 14.973 | -0.929 | 0.893 | 0.294 |
| $Ridge_{\lambda=0.05257397}$ | 18.610 | 0.015 | –0.876 | -0.053 | 0.006 | -2.364 | 0.005 | -0.004 | -13.484 | -0.638 | 0.858 | 0.251 |
| $Ridge_{\lambda=0.6481565}$ | 29.906 | 0.0219 | -0.611 | 0.203 | 0.008 | -1.316 | 0.001 | -0.002 | -25.305 | -0.217 | 0.504 | 0.150 |
| $LASSO_{\lambda=0.01056334}$ | 5.179 | 0 | -0.874 | 0 | 0 | -2.264 | 0.004 | -0.003 | 0 | -0.657 | 0.8122 | 0.270 |
| $LASSO_{\lambda=0.06186968}$ | 3.891 | 0.001 | -0.880 | 0 | 0 | -0.345 | 0 | -0.001 | 0 | -0.164 | 0.385 | 0.243 |

```
y.lm.train = predict(lm.mod, newdata = data.frame(x))
sum((y.lm.train - y)^2)
y.lm.test = predict(lm.mod, newdata = data.frame(x.test))
sum((y.lm.test - y.test)^2)

y.ridge.best.train = predict(ridge.mod.best, newx=x)
sum((y.ridge.best.train - y)^2)
y.ridge.best.test = predict(ridge.mod.best, newx=x.test)
sum((y.ridge.best.test - y.test)^2)

y.ridge.1se.train = predict(ridge.mod.1se, newx=x)
sum((y.ridge.1se.train - y)^2)
y.ridge.1se.test = predict(ridge.mod.1se, newx=x.test)
sum((y.ridge.1se.test - y.test)^2)

y.lasso.best.train = predict(lasso.mod.best, newx=x)
sum((y.lasso.best.train - y)^2)
y.lasso.best.test = predict(lasso.mod.best, newx=x.test)
sum((y.lasso.best.test - y.test)^2)

y.lasso.1se.train = predict(lasso.mod.1se, newx=x)
sum((y.lasso.1se.train - y)^2)
y.lasso.1se.test = predict(lasso.mod.1se, newx=x.test)
sum((y.lasso.1se.test - y.test)^2)
```

Table 4. $RSS$ values for the train ($RSS_{train}$) and the test sets ($RSS_{test}$) for different regression techniques.

| RSS | OLS | $Ridge_{\lambda=0.05257397}$ | $Ridge_{\lambda=0.6481565}$ | $LASSO_{\lambda=0.01056334}$ | $LASSO_{\lambda=0.06186968}$ |
|---|---|---|---|---|---|
| $RSS_{train}$ | 354.7427 | 355.846 | 380.3165 | 356.2117 | 376.4762 |
| $RSS_{test}$ | 318.1009 | 316.3684 | 342.7885 | 315.7833 | 328.3998 |

The error values for all the methods are given in the table 4. Here the smallest RSS value on the train set is predictably achieved by the OLS regression since unlike Ridge and LASSO the OLS does not impose penalties on the coefficients $b_i$. RSS for Ridge and LASSO is again predictably greater when $\lambda$ is selected using the "one-standard-error" rule. It is interesting that the errors on the test set are ordered differently - the minimum is achieved by LASSO at $\lambda = lasso.bestlam$, the second is Ridge at $\lambda = ridge.bestlam$ while OLS takes only the third position. In this case, Ridge and LASSO perform better than OLS on the test set.

## 7. Conclusion

The statistical analysis of the Wine Quality data ([8], [9]) leads to the following conclusions about the OLS, Ridge and LASSO regression procedures.

- The Ridge regression and the LASSO tend to "shrink" to zero coefficient estimates $\widehat{b_i}$ in the sense that they reduce the norm of the estimate vector as $\lambda$ increases.
- The Ridge regression does not produce zero estimates even for large values of $\lambda$.
- The LASSO, unlike Ridge and OLS, performs variable selection i. e. some of the coefficient estimates $\widehat{b_i}$ become exactly equal to zero, which makes the regression model easier to interpret.
- The RSS value calculated on the train set for OLS is less than that for Ridge and LASSO.
- In the considered examples, when $\lambda$ is selected appropriately, the RSS value on the test set for OLS is greater than the RSS for Ridge and LASSO.
- Calculating the RSS on the test data set provides a good way to assess the regression model in contrast to using a single data set.

## Acknowledgements

This work is partially supported by grants of RFBR (project 16-41-630-676 and project 16-01-00184).

## References

[1] X. Yan, X. G. Su, Regression Analysis: Theory and Computing, World Scientific Publishing Co. Pte. Ltd., 2009.
[2] M. H. Kutner, C. J. Nachtsheim, J. Neter, Applied Linear Regression Models, 4th ed., McGraw-Hill Irwin, 2004.
[3] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity. The Lasso and Generalizations, Chapman & Hall, 2015.
[4] B. Efron, T. Hastie, Computer Age Statistical Inference: Algorithms, Evidence and Data Science, Institute of Mathematical Statistics Monographs, 2016.
[5] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2013.
[6] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: Data Mining, Inference, and Prediction, 2nd ed., Springer, 2009.
[7] M. Lichman, UCI Machine Learning Repository, 2013. URL: `http://archive.ics.uci.edu/ml`.
[8] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, UCI Machine Learning Repository — Wine Quality Data Set, 2009. URL: `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`.
[9] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling Wine Preferences by Data Mining from Physicochemical Properties, Decision Support Systems. 47 (2009) 547–553.