Linear statistical models
Academic year 2022-2023

A brief introduction to Ridge regression

Author: Burak Kucuktopal

Contents

Introduction

Regression in Statistics has the objective to describe the effect of a certain variable, called the regressor, on the average of another one, the response variable. To find the adequate model parameters, this comes hand in hand with minimize/maximizing a certain loss function. This function is often used as a metric to quantify the error between the modelled and measured response variable. The loss function for regular linear regression is the Sum of Squared Error(SSE), which takes the sum of the difference between the actual measured data and that estimated by the model. If at each regressor value, there is only one measurement, this difference corresponds to the *bias* of the model at that value.

Minimizing the error is sufficient when we'd like to make assessments about the **same data set**, but once we'd like to make predictions on a **new data set**, the same model could cause some problems because of *overfitting*, which means that the model is too specific and didn't catch the general trend in the bigger population. Therefore, the model at hand must minimize the error, while keeping an eye on the fluctuation in error, also called the variance, through different data sets. In this thesis, Ridge regression will be proposed to find the optimal value at which the variance and the bias are low by adding an extra term to the SSE. The code that was used throughout this paper is open-source and could be consulted on GitHub: `https://github.com/BurakKTopal/Ridge-regression`

1 The problems arising with regular linear regression

In machine learning, when a certain model wants to be trained, the data set gets split into a *train set* and a *test set.* The train set is the set on which the algorithm is 'trained', while the efficiency of it will be checked on the 'unknown' test set.

Suppose we have the following (fictitious) data of the weight(kg) of a person in function of their length(cm):

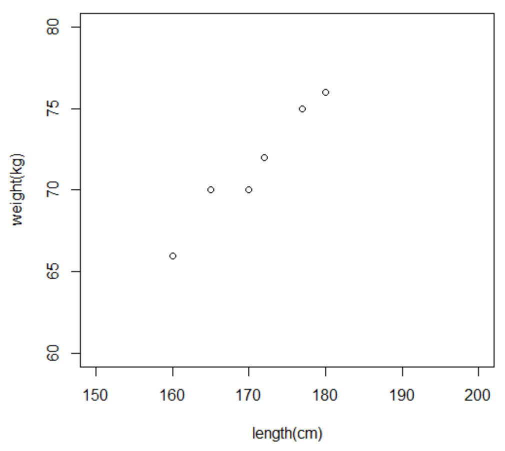


Figure 1: Given data points of weight(in kg) in function of the length(in cm)

The two points in the middle are taken to be the train set, while the others are the test set. Because the train set only contains of two points, the line that will minimize the SSE will be the one through these two:

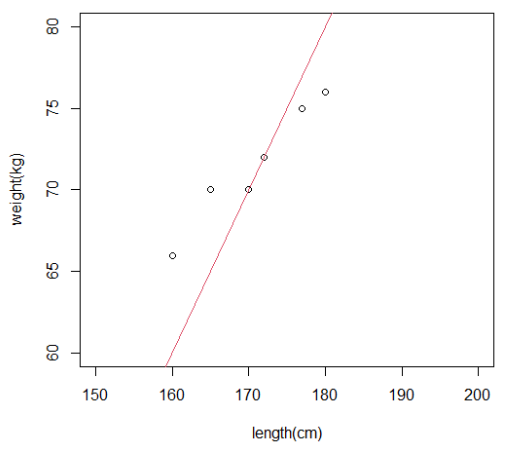


Figure 2: Training set consisting of two points.

Clearly this model isn't as representative for the test set as it is for the train set.

This simple example shows the very big problem in the machine learning: the **bias-variance trade-off**. The bias of the model in the train set is 0(the $SSE = 0$), while it's clearly non-zero for the test set, meaning there is a high variance. The variance of a model is defined as the fluctuations of the bias through different data sets on the same population[1]. Overfitting is the consequence of having a model with a very low bias, but a high variance. Using Ridge regression will guide us to solve this issue.

2 From regular linear regression to Ridge regression

To build up to Ridge regression, it is handy to revisit what the (regular) linear regression says about estimating its parameters.

2.1 Linear regression

The standard linear model, given n observables and p parameters, is described as follows:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1}$$

where:

1. $\mathbf{X}$ is a n x p matrix, also named the *design matrix*. This matrix is given by[3]:
   $$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1(p-1)} \\ 1 & x_{21} & x_{22} & \dots & x_{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{n(p-1)} \end{pmatrix}$$
   With $\mathbf{x}_i^t = (1, x_{i1}, ..., x_{i(p-1)})$ the regressor observations.
2. $\mathbf{Y} \in \mathbb{R}^n$ is the vector containing the n response variables.
3. $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector containing the p parameters of the model.
4. $\boldsymbol{\epsilon} \in \mathbb{R}^n$ with ϵ_i i.i.d. $F_\epsilon(\cdot; \boldsymbol{\nu})$ is the vector containing the n error terms. To account for the underlying assumption of linear regression, one must satisfy: $\mathrm{E}[\epsilon_i] = 0 \wedge \mathrm{var}[\epsilon_i] = \sigma^2$, $i \in \{1, 2, 3, ..., n\}$.

Under the assumptions given above, the estimation $\hat{\boldsymbol{\beta}}$ of the parameter vector $\boldsymbol{\beta}$ is the value as to minimize the SSE[3]:

$$SSE = \|\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|_2^2 \tag{2}$$

with $\|\cdot\|_2$ the L_2-norm. Giving for the estimation of the parameter vector[3]:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y} \tag{3}$$

Keep this in mind, because Ridge regression's will look very similar.

2.2 Ridge Regression

Ridge regression also uses the same model given by formula(1), but with another loss function. The loss function for the Ridge Regression, denoted as RR, is very much related to the SSE and is given by[1]:

$$RR := SSE + \lambda\boldsymbol{\beta}^t\boldsymbol{\beta} := SSE + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{4}$$

Where λ is called the *Ridge parameter* and $\lambda \geq 0$, the term $\lambda\|\boldsymbol{\beta}\|_2^2$ is called the penalty term. The estimation of the parameters is also quite similar to that of the linear regression[2]:

$$\hat{\boldsymbol{\beta}} = [\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^t\mathbf{Y} \tag{5}$$

Where $\mathbf{I}$ is the p x p identity matrix.

Equation (5) could be easily proven. First we rewrite the loss function(suppose $\lambda > 0$):

$$RR(\lambda, \boldsymbol{\beta}) = \text{SSE} + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{6}$$

$$\Longleftrightarrow$$

$$RR(\lambda, \boldsymbol{\beta}) = \mathbf{Y}^t\mathbf{Y} - 2\boldsymbol{\beta}^t\boldsymbol{\beta} + \boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^t\boldsymbol{\beta} \tag{7}$$

To find an extremum of $RR = RR(\lambda, \boldsymbol{\beta})$, we take the partial derivative with respect to $\boldsymbol{\beta}$:

$$\frac{\partial RR(\lambda, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^t\mathbf{Y} + 2\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} + 2\lambda\boldsymbol{\beta} \tag{8}$$

The extremum will be given by equating the partial derivate with respect to $\boldsymbol{\beta}$ to zero:

$$0 = \frac{\partial RR(\lambda, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \tag{9}$$

$$\Longleftrightarrow$$

$$0 = -\mathbf{X}^t\mathbf{Y} + \mathbf{X}^t\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta} = -\mathbf{X}^t\mathbf{Y} + [\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}]\boldsymbol{\beta} \tag{10}$$

The matrix $\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}$ is invertible(see appendix A). So rewriting to $\boldsymbol{\beta} = ...$ gives:

$$\boldsymbol{\beta} = [\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^t\mathbf{Y} \tag{11}$$

Checking if this is a minimum could be done by means of the second-derivative test:

$$\frac{\partial^2 RR(\lambda, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^t} = \frac{\partial(-2\mathbf{X}^t\mathbf{Y} + 2\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} + 2\lambda\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 2\mathbf{X}^t\mathbf{X} + 2\lambda\mathbf{I} \tag{12}$$

The matrix $2\mathbf{X}^t\mathbf{X} + 2\lambda\mathbf{I}$ is positive definite(see appendix A), showing that the extremum value we had found is indeed a minimum.

3 What is the difference between Ridge and linear regression?

If we would change the acquired data a little bit, the minima of each loss function will differ a little bit too. But with Ridge regression there is more stability compared to that of the linear one due to the penalty term due to the fact that it gives an extra "boundedness" of the loss function. This penalty term decreases the effect of the parameters on the response variables. In the extreme case, where $\lambda \to \infty$, the $\mathbf{X}^t\mathbf{X}$ in equation (5) can be neglected, giving for the estimation:

$$\hat{\boldsymbol{\beta}} = [\lambda\mathbf{I}]^{-1}\mathbf{X}^t\mathbf{Y} = \frac{\mathbf{X}^t\mathbf{Y}}{\lambda} \tag{13}$$

So the $\hat{\beta}_i$'s go to zero. In this case, the variance is very low, but now the bias is very high because we'd only have a zeroth degree approximation.

We could also visualize the stability of the Ridge regression by changing two data points(see table 1 in appendix B) and plot the graphs given by the Ridge and regular linear regression model estimations:

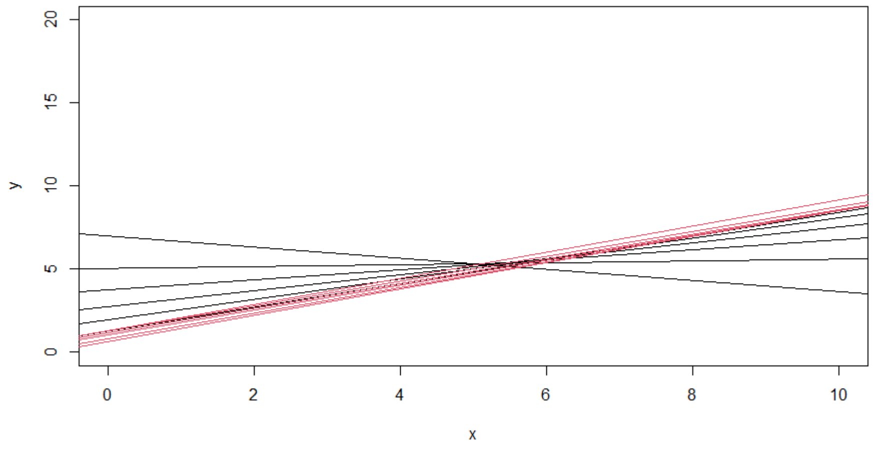


Figure 3: Black: plot of estimated model using linear regression. Red: plot of estimated model using Ridge regression($\lambda = 1$).

We observe that the variability of parameters obtained by Ridge regression, isn't as much as with the linear regression when changing the data points' coordinates. This shows that Ridge regression can be better to use when we'd like to have more stable results over different data sets on the same population, countering high variability. But this lower variability does come with a higher bias on the used data set. The question arises: what is the best value for λ such that the total of variance and bias is as low as possible?

Note that to penalize the SSE, we can't take too low values because of possible overfitting, while too high values give no variability at all. The exact method, together with its theoretical background, will be presented in section 5.

4 The magic behind Ridge regression

From linear regression, it's a necessity to have for p parameters to estimate, at least p observations. Because, take equation (3):

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y} \tag{14}$$

Suppose we have n observations for which: $n < p$. There holds for the rank of X[4]:

$$rank(\mathbf{X}) \leq min\{n, p\} = n < p \tag{15}$$

For the transpose of $\mathbf{X}$, the same thing applies:

$$rank(\mathbf{X}^t) \leq min\{n, p\} = n < p \tag{16}$$

For the rank of the product of two matrices[4] following equation is valid:

$$ranks(\mathbf{X}^t\mathbf{X}) \leq min\{rank(\mathbf{X}^t), rank(\mathbf{X})\} = n < p \tag{17}$$

Therefore[4]:

$$Det(\mathbf{X}^t\mathbf{X}) = 0 \tag{18}$$

So when we're working with linear regression, we can't compute uniquely the estimations of $\boldsymbol{\beta}$ for $n < p$.

For Ridge regression the matrix $\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}$ is invertible for $\lambda > 0$(see appendix A), so there can be made an estimation for β. For example, take the following model for which we need to estimate $\boldsymbol{\beta}^t = (\beta_0, \beta_1)$:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{19}$$

With ϵ_i i.i.d. $F_\epsilon(\cdot; \boldsymbol{\nu})$

Suppose only one observation (x_1, y_1) is given. In linear regression, we couldn't do any estimation because the matrix $\mathbf{X}^t\mathbf{X}$ isn't invertible. Geometrically it means that we could draw infinitely many lines through one given point, because for all of these lines $SSE = 0$. But in Ridge regression, there is no restriction and even one observation suffices to make an estimation. This can be visualized using the data set introduced in the section 1:

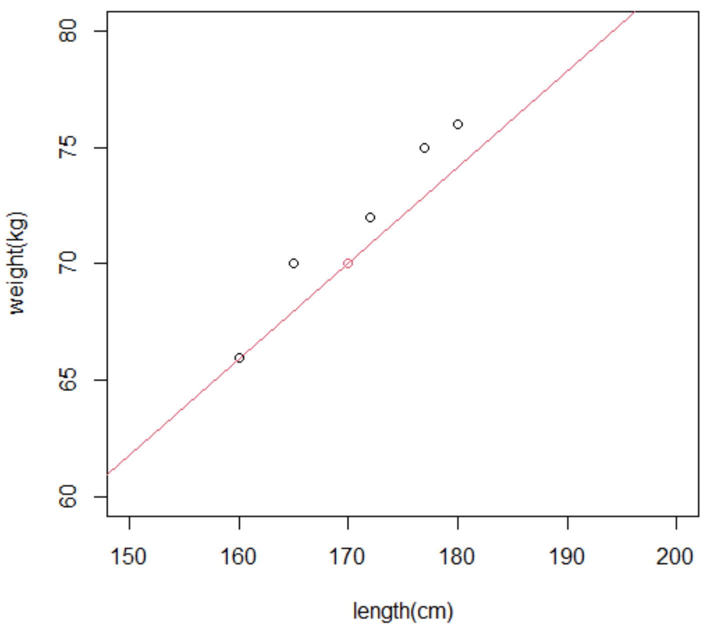


Figure 4: Plot of regression line with $\lambda = 1$ and train set= $\{(170{,}70)\}$(train set represented by red dot).

We clearly observe that even with a training set with only one point, we still can get some decent results. Before concluding with a real data set analysis, the method to optimize the λ parameter will be shown.

5 Theoretical background on finding the optimal value for λ

In the example discussed in section 4 the lambda parameter was arbitrarily chosen to be 1, for demonstration purposes. But is this value the ideal one? You could question why we couldn't simply vary lambda and look for the one which gives the lowest error in the test set. But in this case the test set would also be used as part of the training set, by which the algorithm cannot be appropriately evaluated on the presence of overfitting and the amount of error. To catch this general trend of the population, we we'll use **k-fold cross validation**, in particular $k = 10$.

5.1 10-fold cross validation

Suppose we have a data set X given. To train our algorithm, we would divide this data set into a training set, X_{train}, and a test set, X_{test}. Training means in this setting that we estimate the parameters given by equation (11) for a given λ. As stated above, one couple of training-test set isn't enough to get the underlying trend of the population. But there is a little trick to get an idea of the the general structure of our population; we partition our data set X into $k = 10$ different subsets, $V_\alpha \neq \emptyset$, with α in the index set $\mathbb{A} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, obeying following conditions:

1. $\bigcup_{\alpha \in \mathbb{A}} V_\alpha = X$
2. $V_i \cap V_j = \emptyset, \forall i \neq j$

This partition is used to create a new test set, $Y_{test} = V_i$ for a certain $i \in \mathbb{A}$ and test set $Y_{train} = X \backslash Y_{test}$. With the 10 $V_i's$, this can be done in 10 different ways. For each $Y_{train,j}$, $j \in \mathbb{A}$, there will be estimated the p-parameters$(\beta_{0,j}, \beta_{1,j}, ..., \beta_{(p-1),j})$, using equation (11). Once the parameter values are calculated, we test it on the $Y_{test,j} = \{y_{1,j}, y_{2,j}, ..., y_{r,j}\}$ with $r = \frac{|X|}{10}$ and calculate the residual error for each data element corresponding to the (multidimensional) i-th regressor, $\vec{x}_{i,j} = (x_{0i,j}, x_{1i,j}, ..., x_{(p-1)i,j})$, of the j-th partition[5]:

$$Err_{i,j}(\lambda, \vec{x}_{i,j}) = y_{i,j} - \sum_{i=0}^{p-1} \beta_{l,j}(\lambda) x_{li,j} \tag{20}$$

Leading to the residual error of the j-th partition:

$$Residual_j(\lambda) = \frac{1}{r} \sum_{i=1}^{r} [Err_{i,j}(\lambda, \vec{x}_{i,j})]^2 \tag{21}$$

This procedure is done for the other 9 possible partitions, whereafter each error is being squared and summed up:

$$\Lambda(\lambda) := Residual(\lambda) = \sum_{i=1}^{10} Residual_j(\lambda) \tag{22}$$

The last step is to find λ that minimizes $\Lambda(\lambda)$.

This method intuitively means that we are taking the one data set and make 'new' data sets out of it, as if we really had 10 different data sets and not one big one. Making these partitions shows us how different values of λ effect the total residual error. The minimum of Λ corresponds to the value of the penalty term at which the sum of the residual errors of the differently mutated data set is minimal, avoiding a too high variance and obtaining a more general picture of the population.

6 Toxicogenomics and Ridge regression

Now that we have the basis to understand Ridge regression, we will give a final, real-world example and dive deep into its analysis. We'll discuss the results with a randomly chosen $\lambda = 1$, and after this the optimized one to show the difference.

6.1 Problem

To determine the toxicity of a chemical substance, the classical approach is to surround liver cells by these elements and look at the reaction of the cells. This process is very time-consuming and expensive.

Therefore people hope to find a connection between gene expression of liver cells and toxicity. The experimental setup is as follows: liver cells are exposed to 30 chemical substances. There are two given data sets: one of the expression of 4000 genes, labeled as the data set 'X', while the data set 'Y' represents the toxicity score of the 30 chemical substances found by means of the classical, time-consuming and expensive way.

In this study, we'll validate whether the following question holds:

Is there a linear correspondence between the gene expression of the liver cells and the toxicity of the surrounding chemicals?

6.2 Analysis of the data

The number of observations, n, are 30 and the parameters we need to estimated, p, is 4001: one for every gene and one for the intercept. Since $n < p$, we can't use linear regression(see section 4), but we can use Ridge regression to fit the linear regression model given by equation (1). We could manually plug in the values in equation (11), but R has a function to do it for us from the *glmnet* package. Constituting the model(See the GitHub link from the introduction) and plotting the values of the parameters in function of their corresponding gene number:

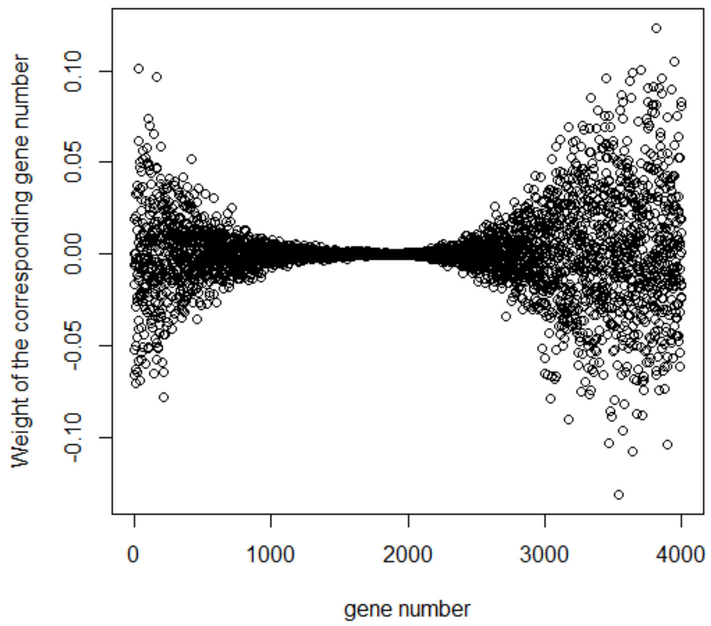


Figure 5: Plot of values of the parameters of the model in function of their gene number.

From figure 5 we see that for the genes from 1200-2300 there is a negligible influence on the toxicity. When we go to the extremes of the spectrum, there is starting to be a dependency to the point where the order of the coefficients of the genes is 0.1. This can be quantified using the Sum of Squares Regression(SSR), defined as[3]:

$$SSR := \sum_{i=1}^{30} (\hat{Y}_i - \bar{Y})^2 \tag{23}$$

Calculating this for the estimated model gives $SSR = 96.85 \gg 0$. This means that there is a clear effect of the genes(=regressors) on the toxicity. To check that this effect is linear, we can calculate the *coefficient of determination*, R^2. For this, we can use the identity[3]:

$$R^2 = 1 - \frac{SSE}{SSTot} \tag{24}$$

With SST the 'Sum of Squares Total' and SSE given by respectively:

$$\sum_{i=1}^{30} (Y_i - \bar{Y})^2 \wedge \sum_{i=1}^{30} (\hat{Y}_i - Y_i)^2$$

Computing using the R-code:

$$SST = 98.99 \wedge SSE = 0.0135$$

Thus giving $R^2 = 1 - \frac{0.0135}{98.99} = 0.99986$, concluding that the variability in the data is almost perfectly explained by the linear model. More specifically; 99,986% of the variability of the observed toxicity is explained because the (average) toxicity value is linearly dependent on the (4000) genes. We therefore can conclude with high certainty that the toxicity of the chemical substances is linearly dependent on the gene expression of the liver cells surrounded by these substances.

6.3 Finding the optimal value for the penalty term

In the previous section, λ was chosen arbitrarily chosen to be 1, but does this value give results in a model that has a low bias and variance? To find the optimal value, we run our code and get following log-scaled plot of $\Lambda(\lambda)$:

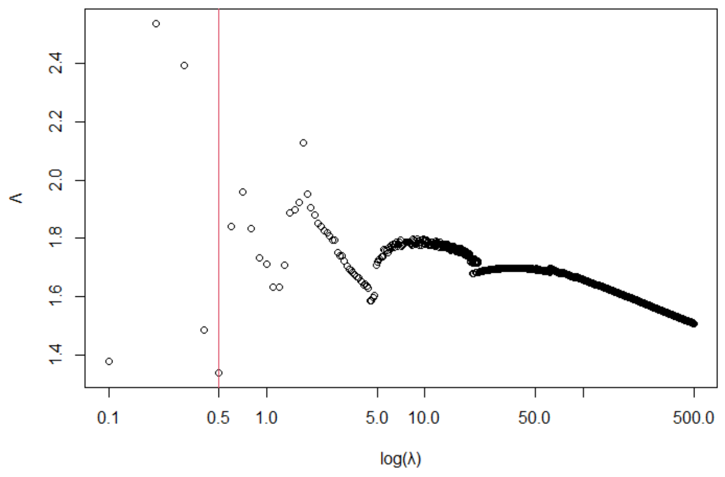


Figure 6: Log scale of $\Lambda(\lambda)$, with indication of minimum(vertical red line).

So the value at which $\Lambda(\lambda)$ is minimized is $\lambda = 0.5$.

6.3.1 Revisiting the data-analysis of section 6.2 with $\lambda = \frac{1}{2}$

With the optimal penalty term, the plot of the value of the parameters in function of their corresponding gene number is as follows:

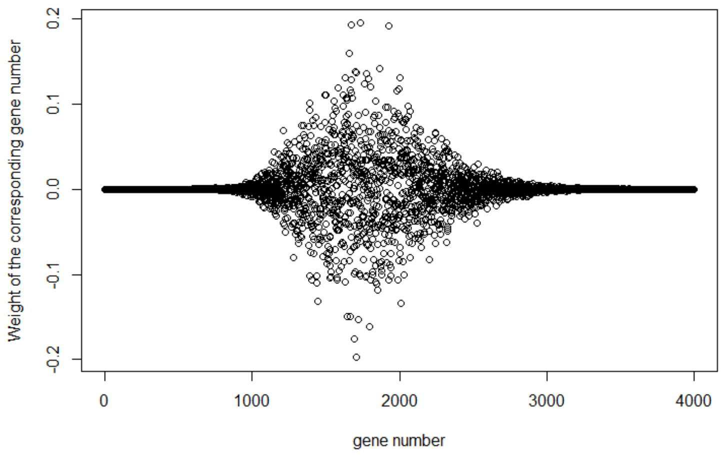


Figure 7: Plot of value of the parameters of the model in function of their gene number, with optimal lambda value($\lambda = \frac{1}{2}$).

This plot is almost the inverse of that of figure 5, but according to the 10-fold cross validation(see section 5.1), this model is the one which gives the lowest total sum of residuals over the 10 different partitions.

Comparing the SSR, SSE and R^2 between the two different $\lambda's$:

	SSE	SST	R^2
$\lambda = 1$	$1.35 \cdot 10^{-2}$	98.990	0.99986
$\lambda = \frac{1}{2}$	$6.266 \cdot 10^{-11}$	98.994	≈ 1

We see that the coefficient of determination is even higher in this case. Together with avoidance of overfitting due to the principles of 10-fold cross validation, we can safely assume that the model at hand is better than that presented at section 6.2. This model reflects a (strong) linear connection between the two variables. We therefore conclude:

The toxicity of the chemical substances presented in the data set has a strong linear relationship with the gene expressions of the liver cells.

7 Conclusion

Ridge regression was introduced to cover up for the overfitting and instability that could sometimes happen in regular linear regression by adding a penalty term $\lambda\|\boldsymbol{\beta}\|_2^2$ to its loss function. The Ridge parameter λ in the penalty term serves as the medium to control the amount of variance of the model; $\lambda = 0$ gives back estimations of regular linear regression, while $\lambda \to \infty$ gives $\hat{\boldsymbol{\beta}} \to 0$, giving no valuable model estimation at all. Therefore, we concluded that this parameter must be treated with care and choosing some arbitrary value, could do more harm than good.

To find the optimal value, we introduced the method of 10-fold cross validation, which 'creates' data sets by mutating different subsets of one particular data set into different partitions. By minimizing the function $\Lambda(\lambda)$, defined as the sum of residual of these 10 different partition, $\lambda_{optimal}$ can be found. Intuitively, the argument of the minimum returns the λ value at which there is the most consistency of model predictions throughout different data sets of the same population.

Another credible feature, besides the consistent model predictions, of Ridge regression is that there is no restriction on the number of observations. In contrast, regular linear regression required at least p observation for estimating p parameters. This had to do with the mathematical observation that the p x p matrix $\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I_p}$ is only invertible for $n \geq p$ if $\lambda = 0$ and $\forall n \in \mathbb{N}_0$ if $\lambda > 0$. This led to the use of Ridge regression on a real-world example covering the possible linear correspondence between the expression of 4000 genes and the toxicity score of 30 chemical substances by only making use of 30 observations. Using Ridge regression with the Ridge parameter $\lambda_{optimal} = \frac{1}{2}$ and using the appropriate metrics to validate on linearity, we concluded that there was a strong linear effect of the gene expression of the liver cells, surrounded by the 30 chemicals, on the toxicity score of these substances.

8 Appendices

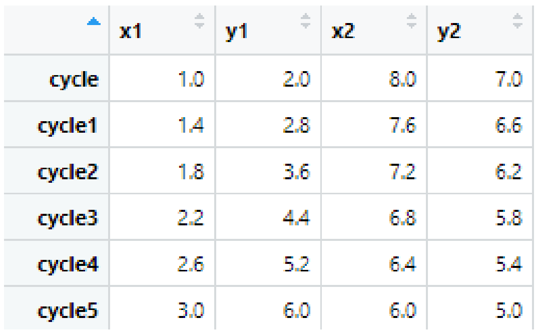
8.1 Appendix A: proof of invertibility of $\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}$

Suppose $\lambda > 0$, it suffices to show that the matrix is positive definite[6]: Take $\mathbf{y} \in \mathbb{R}^p \setminus \{0\}$:

$$\mathbf{y}^t(\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I})\mathbf{y} = (\mathbf{X}\mathbf{y})^t(\mathbf{X}\mathbf{y}) + \lambda\mathbf{y}^t\mathbf{y} = \|\mathbf{X}\mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|_2^2$$

Because a norm is always positive, and the Ridge parameter is strictly positive, $\mathbf{X}^t\mathbf{X} + \lambda\mathbf{I}$ is strictly positive and by definition positive-definite, thus invertible $\square$

8.2 Appendix B: table 1

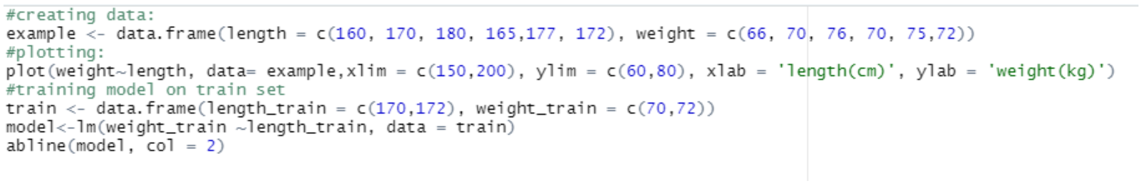


	x1	y1	x2	y2
cycle	1.0	2.0	8.0	7.0
cycle1	1.4	2.8	7.6	6.6
cycle2	1.8	3.6	7.2	6.2
cycle3	2.2	4.4	6.8	5.8
cycle4	2.6	5.2	6.4	5.4
cycle5	3.0	6.0	6.0	5.0

Table 1: Table with changing of the two data points: one with coordinates (x1, y1), and the other (x2, y2). Cycles represents the coordinates of the data points in one for loop, see 9

8.3 Appendix C: R code

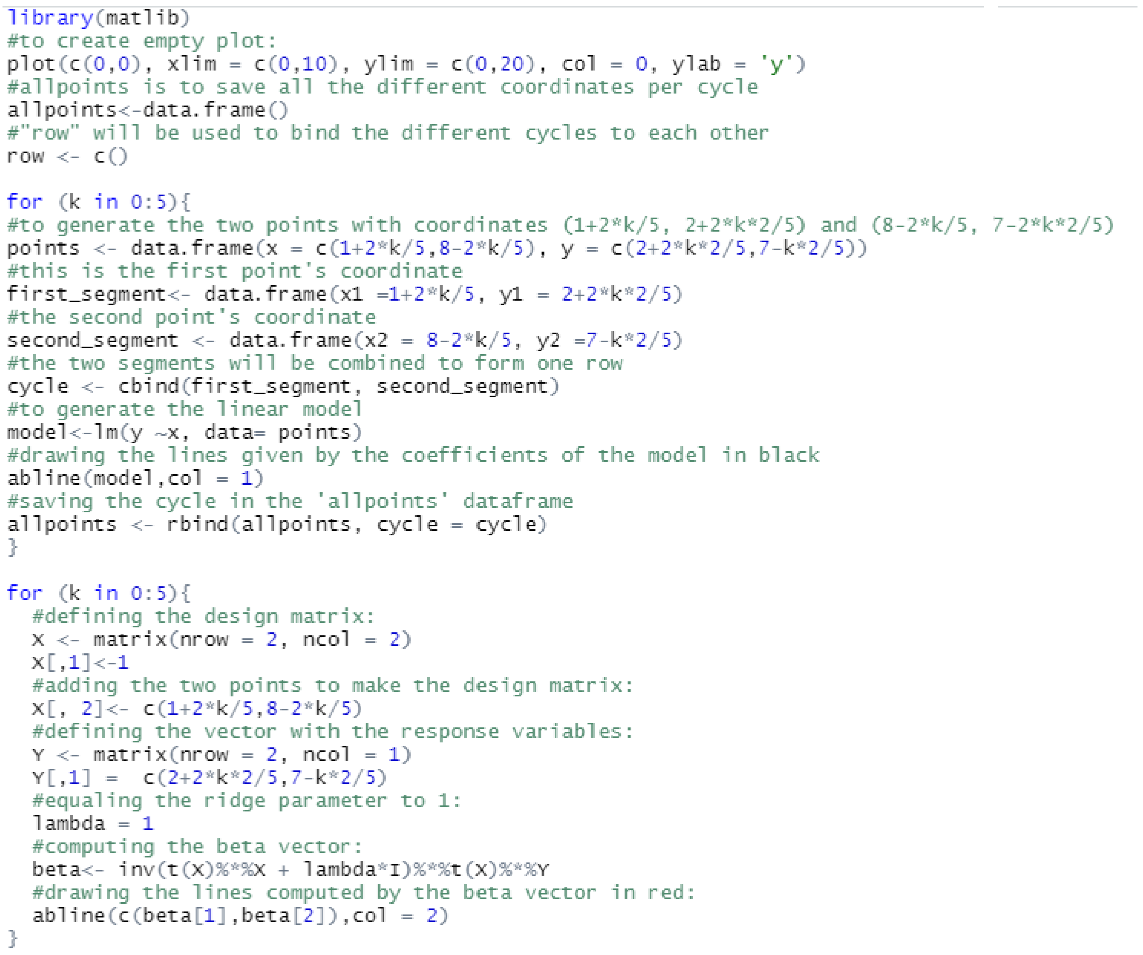
8.3.1 code for section 1



```r
#creating data:
example <- data.frame(length = c(160, 170, 180, 165,177, 172), weight = c(66, 70, 76, 70, 75,72))
#plotting:
plot(weight~length, data= example,xlim = c(150,200), ylim = c(60,80), xlab = 'length(cm)', ylab = 'weight(kg)')
#training model on train set
train <- data.frame(length_train = c(170,172), weight_train = c(70,72))
model<-lm(weight_train ~length_train, data = train)
abline(model, col = 2)
```

Figure 8: Code to compute 2

8.3.2 Code for section 3



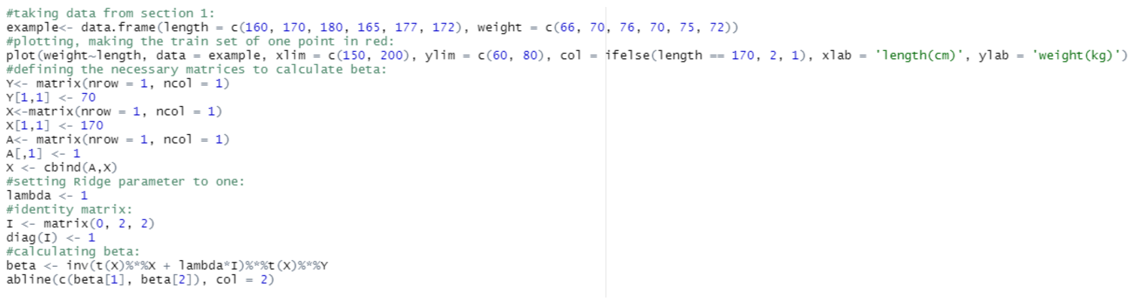
```
library(matlib)
#to create empty plot:
plot(c(0,0), xlim = c(0,10), ylim = c(0,20), col = 0, ylab = 'y')
#allpoints is to save all the different coordinates per cycle
allpoints<-data.frame()
#"row" will be used to bind the different cycles to each other
row <- c()

for (k in 0:5){
#to generate the two points with coordinates (1+2*k/5, 2+2*k*2/5) and (8-2*k/5, 7-2*k*2/5)
points <- data.frame(x = c(1+2*k/5,8-2*k/5), y = c(2+2*k*2/5,7-k*2/5))
#this is the first point's coordinate
first_segment<- data.frame(x1 =1+2*k/5, y1 = 2+2*k*2/5)
#the second point's coordinate
second_segment <- data.frame(x2 = 8-2*k/5, y2 =7-k*2/5)
#the two segments will be combined to form one row
cycle <- cbind(first_segment, second_segment)
#to generate the linear model
model<-lm(y ~x, data= points)
#drawing the lines given by the coefficients of the model in black
abline(model,col = 1)
#saving the cycle in the 'allpoints' dataframe
allpoints <- rbind(allpoints, cycle = cycle)
}

for (k in 0:5){
  #defining the design matrix:
  X <- matrix(nrow = 2, ncol = 2)
  X[,1]<-1
  #adding the two points to make the design matrix:
  X[, 2]<- c(1+2*k/5,8-2*k/5)
  #defining the vector with the response variables:
  Y <- matrix(nrow = 2, ncol = 1)
  Y[,1] =  c(2+2*k*2/5,7-k*2/5)
  #equaling the ridge parameter to 1:
  lambda = 1
  #computing the beta vector:
  beta<- inv(t(X)%*%X + lambda*I)%*%t(X)%*%Y
  #drawing the lines computed by the beta vector in red:
  abline(c(beta[1],beta[2]),col = 2)
}
```

Figure 9: Code to compute 3

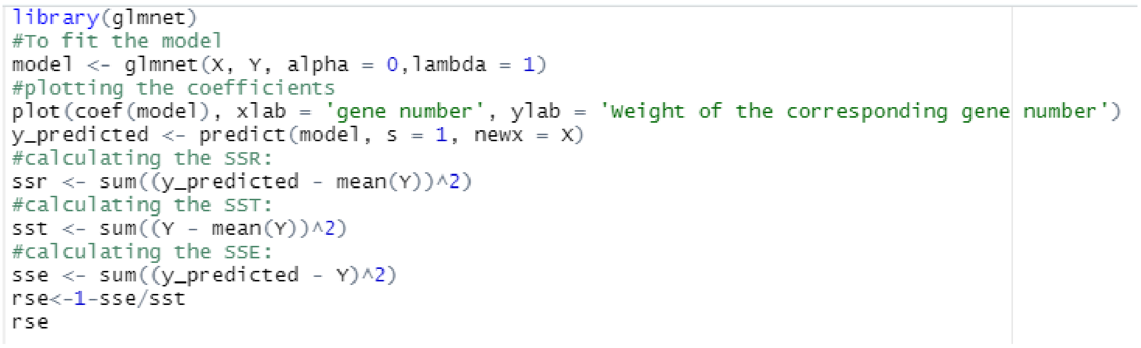
8.3.3 Code for section 4



```
#taking data from section 1:
example<- data.frame(length = c(160, 170, 180, 165, 177, 172), weight = c(66, 70, 76, 70, 75, 72))
#plotting, making the train set of one point in red:
plot(weight~length, data = example, xlim = c(150, 200), ylim = c(60, 80), col = ifelse(length == 170, 2, 1), xlab = 'length(cm)', ylab = 'weight(kg)')
#defining the necessary matrices to calculate beta:
Y<- matrix(nrow = 1, ncol = 1)
Y[1,1] <- 70
X<-matrix(nrow = 1, ncol = 1)
X[1,1] <- 170
A<- matrix(nrow = 1, ncol = 1)
A[,1] <- 1
X <- cbind(A,X)
#setting Ridge parameter to one:
lambda <- 1
#identity matrix:
I <- matrix(0, 2, 2)
diag(I) <- 1
#calculating beta:
beta <- inv(t(X)%*%X + lambda*I)%*%t(X)%*%Y
abline(c(beta[1], beta[2]), col = 2)
```

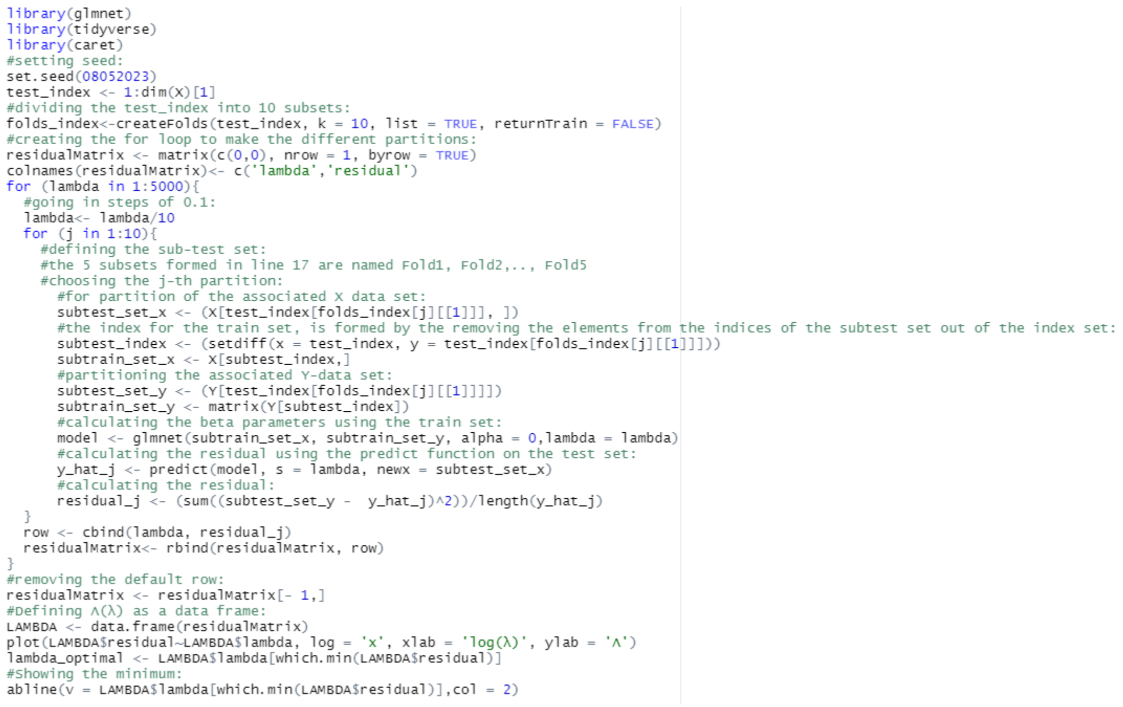
Figure 10: Code to compute 4

8.3.4 Code for section 6



```
library(glmnet)
#To fit the model
model <- glmnet(X, Y, alpha = 0,lambda = 1)
#plotting the coefficients
plot(coef(model), xlab = 'gene number', ylab = 'Weight of the corresponding gene number')
y_predicted <- predict(model, s = 1, newx = X)
#calculating the SSR:
ssr <- sum((y_predicted - mean(Y))^2)
#calculating the SST:
sst <- sum((Y - mean(Y))^2)
#calculating the SSE:
sse <- sum((y_predicted - Y)^2)
rse<-1-sse/sst
rse
```

Figure 11: Code to compute 5



```
library(glmnet)
library(tidyverse)
library(caret)
#setting seed:
set.seed(08052023)
test_index <- 1:dim(X)[1]
#dividing the test_index into 10 subsets:
folds_index<-createFolds(test_index, k = 10, list = TRUE, returnTrain = FALSE)
#creating the for loop to make the different partitions:
residualMatrix <- matrix(c(0,0), nrow = 1, byrow = TRUE)
colnames(residualMatrix)<- c('lambda','residual')
for (lambda in 1:5000){
  #going in steps of 0.1:
  lambda<- lambda/10
  for (j in 1:10){
    #defining the sub-test set:
    #the 5 subsets formed in line 17 are named Fold1, Fold2,.., Fold5
    #choosing the j-th partition:
      #for partition of the associated X data set:
      subtest_set_x <- (X[test_index[folds_index[j][[1]]], ])
      #the index for the train set, is formed by the removing the elements from the indices of the subtest set out of the index set:
      subtest_index <- (setdiff(x = test_index, y = test_index[folds_index[j][[1]]]))
      subtrain_set_x <- X[subtest_index,]
      #partitioning the associated Y-data set:
      subtest_set_y <- (Y[test_index[folds_index[j][[1]]]])
      subtrain_set_y <- matrix(Y[subtest_index])
      #calculating the beta parameters using the train set:
      model <- glmnet(subtrain_set_x, subtrain_set_y, alpha = 0,lambda = lambda)
      #calculating the residual using the predict function on the test set:
      y_hat_j <- predict(model, s = lambda, newx = subtest_set_x)
      #calculating the residual:
      residual_j <- (sum((subtest_set_y -  y_hat_j)^2))/length(y_hat_j)
  }
  row <- cbind(lambda, residual_j)
  residualMatrix<- rbind(residualMatrix, row)
}
#removing the default row:
residualMatrix <- residualMatrix[- 1,]
#Defining Λ(λ) as a data frame:
LAMBDA <- data.frame(residualMatrix)
plot(LAMBDA$residual~LAMBDA$lambda, log = 'x', xlab = 'log(λ)', ylab = 'Λ')
lambda_optimal <- LAMBDA$lambda[which.min(LAMBDA$residual)]
#Showing the minimum:
abline(v = LAMBDA$lambda[which.min(LAMBDA$residual)],col = 2)
```

Figure 12: Code to compute 6 and 7

9 Literature

- [1]https://stats.stackexchange.com/questions/522829/what-is-meant-by-low-bias-and-high-var
- [2]https://machinelearningcompass.com/machine_learning_models/ridge_regression/
- [3] Universiteit Hasselt. Lineaire en statistische modellen cursusnota's 2022-2023
- [4] https://en.wikipedia.org/wiki/Rank_(linear_algebra)
- [5] https://www.statology.org/ridge-regression-in-r/
- [6] https://math.stackexchange.com/questions/1059566/explain-proof-that-any-positive-defini
- Github link to play around with the code: https://github.com/BurakKTopal/Ridge-regression