Chapter 3
Elements for Building Supervised Statistical Machine Learning Models

3.1 Definition of a Linear Multiple Regression Model

A linear multiple regression model (LMRM) is a useful tool for investigating linear relationships between two or more explanatory variables (inputs, features in machine learning literature) (X) and the conditional expected value of a response $E(Y/X)$. Due to its simplicity, adequate fitting, and easily interpretable results, this has been one of the most popular techniques for studying the association between variables. Specifically, regarding the latter task, this is a useful approach and an ideal (natural) starting point for studying more advanced methods (James et al. 2013) of association and prediction.

In this chapter, we review the main concepts and approaches for fitting a linear regression model.

3.2 Fitting a Linear Multiple Regression Model via the Ordinary Least Square (OLS) Method

In a general context, we have a covariate vector $X = (X_1, \ldots, X_p)^{\mathrm{T}}$ and we want to use this information to predict or explain how this variable affects a real-value response Y. The linear multiple regression model assumes a relationship given by

$$Y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \epsilon, \tag{3.1}$$

where ϵ is a random error with mean 0, $E(\epsilon) = 0$ and is independent of X. This error is included in the model to capture measurement errors and the effects of other unregistered explanatory variables that can help to explain the mean response.

© The Author(s) 2022

O. A. Montesinos López et al., *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, https://doi.org/10.1007/978-3-030-89010-0_3

Then, the conditional mean of this model is $E(Y|X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$ and the conditional distribution of Y given X is only affected by the information of X.

For estimating the parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^{\mathrm{T}}$, usually we have a set of data $\left(\boldsymbol{x_i}^{\mathrm{T}}, y_i\right)$, $i = 1, \ldots, n$, often known as training data, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$ is a vector of features measurement and y_i is the response measurement corresponding to the ith individual drawn. The most common method for estimating $\boldsymbol{\beta}$ is the least squares method (OLS) that consists of taking the $\boldsymbol{\beta}$ value that minimizes the residual sum of squares defined as

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{\beta}_0\right)^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}),$$

where $\boldsymbol{\beta}_0 = (\beta_1, \ldots, \beta_p)^{\mathrm{T}}$, $\boldsymbol{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ is the vector with the response values of all individuals, and $\boldsymbol{X}$ is an $n \times (p + 1)$ matrix that contains the information of the measured features of all individuals, including the intercept in the first entry:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}.$$

If the $\boldsymbol{X}$ matrix has full column rank, then by differentiating the residual sum of squares with respect to the $\boldsymbol{\beta}$ coefficients, we can find the set of $\boldsymbol{\beta}$ parameters that minimize the $\mathrm{RSS}(\boldsymbol{\beta})$,

$$\frac{\mathrm{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{2y}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^{\mathrm{T}}\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)\boldsymbol{\beta}}{\partial \boldsymbol{\beta}} = \boldsymbol{2}\left[\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)\boldsymbol{\beta} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}\right]$$

This derivative is also known as the gradient of the residual sum of squares. Then by setting the gradient of the residual sum of squares to zero, we obtain the normal equations

$$\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)\boldsymbol{\beta} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}$$

The solution to the normal equations is unique and gives the OLS estimator of $\boldsymbol{\beta}$

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y},$$

where super index -1 indicates the inversion matrix.

From the above assumptions, we can show that this estimator is unbiased

$$\begin{aligned} E\left(\widehat{\boldsymbol{\beta}}\right) &= E\left[\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right] = E\left[\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{\beta} + \epsilon)\right] \\ &= E\left[\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta}\right] + \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}E(\epsilon) = \boldsymbol{\beta}. \end{aligned}$$

and with the additional assumption that the observation responses y_i's are uncorrelated and have the same variance, $\text{Var}(y_i) = \sigma^2$, we can also show that the variance–covariance matrix of this is

$$\text{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \sigma^2 \left(\boldsymbol{X}^{\text{T}}\boldsymbol{X}\right)^{-1}.$$

When the input features only contain the information of a variable ($p = 1$), the resulting model is known as simple linear regression and can be easily visualized in the Cartesian plane. When $p = 2$, the above multiple linear regression describes a plane in the three-dimensional space (x_1, x_2, y). In general, the conditional expected value of this model defines a hyperplane in the p-dimensional space of the input variables (Montgomery et al. 2012).

The fitted values corresponding to all the training individuals are

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}\left(\boldsymbol{X}^{\text{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\text{T}}\boldsymbol{y} = \boldsymbol{H}\boldsymbol{y},$$

where the matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^{\text{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{T}}$ is commonly called the hat matrix. This is because the vector of the observed response values is mapped by this expression to a vector of fitted values (Montgomery et al. 2012), in this way, puts the hat on $\boldsymbol{y}$ (Hastie et al. 2009). In a similar way, a predicted value of an arbitrary individual with feature $\boldsymbol{x}$ can be obtained by

$$\widehat{\boldsymbol{y}}^{*} = \boldsymbol{x}^{*\text{T}}\widehat{\boldsymbol{\beta}},$$

where $\boldsymbol{x}^{*} = (1, \boldsymbol{x}^{\text{T}})^{\text{T}}$.

An unbiased estimator for the common residual variance σ^2 is obtained by

$$\begin{aligned}
\widehat{\sigma}^2 &= \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \\
&= \frac{1}{n-p-1}\sum_{i=1}^{n} e_i^2 \\
&= \frac{1}{n-p-1}\boldsymbol{e}^{\text{T}}\boldsymbol{e} \\
&= \frac{1}{n-p-1}(\boldsymbol{y} - \widehat{\boldsymbol{y}})^{\text{T}}(\boldsymbol{y} - \widehat{\boldsymbol{y}}) \\
&= \frac{1}{n-p-1}\boldsymbol{y}^{\text{T}}(\boldsymbol{I}_n - \boldsymbol{H})\boldsymbol{y},
\end{aligned}$$

where $e_i = y_i - \widehat{y}_i$ is known as the residual of the model corresponding to the individual i, $\boldsymbol{e} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$ is the vector of all residual values, and $\boldsymbol{I}_n$ is the identity matrix of order $n \times n$.

The traditional inferential and prediction analysis for this model assumes that the random error ϵ is normally distributed with mean zero and variance σ^2. With this we

can show that the OLS of beta coefficients, $\widehat{\boldsymbol{\beta}}$, is a random vector distributed according to a multivariate normal distribution with vector mean $\boldsymbol{\beta}$ and a variance–covariance matrix, as previously defined (Montgomery et al. 2012; Hastie et al. 2009; Rencher and Schaalje 2008). Another important fact that will be described in more detail in the next section, is that under the Gaussian assumption over errors, the OLS of $\boldsymbol{\beta}$ coincides with the maximum likelihood estimator.

We can also show that $(n-p-1)\widehat{\sigma}^2/\sigma^2$ is independent of $\widehat{\boldsymbol{\beta}}$ and distributed according to a Chi-squared distribution with $n-p-1$ degrees of freedom. Based on this and on the properties of the normal and t-student distributions, we show that for each $j=0,\ldots,p$, $T_j = \left(\widehat{\beta}_j - \beta_j\right)/\sqrt{c_{jj}\widehat{\sigma}^2}$, where c_{jj} is the $(j+1, j+1)$ elements of the matrix $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}$, are random variables with a t-student distribution with $n-p-1$ degrees of freedom (t_{n-p-1}). That is, $T_j \sim t_{n-p-1}$ and $\sim$ stands for distributed as. From here, a $100(1-\alpha)\%$ confidence interval for a particular beta coefficient, β_j, is given by

$$\widehat{\beta}_j \pm t_{1-\alpha/2,\ n-p-1}\sqrt{c_{jj}\widehat{\sigma}^2},$$

where $t_{\alpha,\, n-p-1}$ is the α quantile of the t-student distribution with $n-p-1$ degrees of freedom. Similarly, a $100(1-\alpha)\%$ joint confidence region for all the beta coefficients, $\boldsymbol{\beta}$, is given if these values satisfy

$$\frac{\left(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right)^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\left(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right)}{(p+1)\widehat{\sigma}^2} \leq F^{p+1}_{1-\alpha,\ n-p-1},$$

where $F^{p+1}_{\alpha,n-p-1}$ denotes the α quantile of the F distribution with $p+1$ and $n-p-1$ degrees of freedom in the numerator and denominator, respectively (Rencher and Schaalje 2008).

In a similar way, to test a hypothesis over a specific beta coefficient, $H_{0j}=\beta_j=\beta_{j0}$, the following rule can be used: reject H_{0j} if $T_{j0} = \left(\widehat{\beta}_j - \beta_{j0}\right)/\sqrt{c_{jj}\widehat{\sigma}^2}$ is "large" in magnitude, that is, if $|T_{j0}| > t_{1-\alpha/2,\, n-p-1}$, where α is the desired level test. More generally, the test $H_0 = \boldsymbol{W\beta} = \boldsymbol{w}$, where $\boldsymbol{W}$ is a $q \times (p+1)$ matrix of rank $q \leq p+1$, can be performed using the following rule:

$$\text{reject } H_0 \text{ if } F = \frac{n-p-1}{q}\,\frac{(\boldsymbol{W\beta}-\boldsymbol{w})^{\mathrm{T}}\left[\boldsymbol{W}\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{W}^{\mathrm{T}}\right]^{-1}(\boldsymbol{W\beta}-\boldsymbol{w})}{\widehat{\sigma}^2} \geq F^{p+1}_{1-\alpha,n-p-1}.$$

3.3 Fitting the Linear Multiple Regression Model via the Maximum Likelihood (ML) Method

The maximum likelihood (ML) estimation is a more general and popular method for estimating the parameters of a model (Casella and Berger 2002). It consists of finding the parameter value that maximizes the "probability" of observed values in the sample under the adopted model. Specifically, if $\left(\boldsymbol{x}_i^{\mathrm{T}}, y_i\right)$, $i = 1, \ldots, n$, is a set of observations from a multiple linear regression model (3.1) with homoscedastic and uncorrelated errors, the MLE of $\boldsymbol{\beta}$ and σ^2, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$, of this model is defined as

$$\left(\widehat{\boldsymbol{\beta}}^{\mathrm{T}}, \widehat{\sigma}^2\right) = \arg\max_{\boldsymbol{\beta}, \sigma^2} L\left(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right),$$

where $L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X})$ is the likelihood function of the parameters, which is the probability of the observed response values but viewed as a function of the parameters

$$L\left(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right].$$

Then, the $\log(L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}))$ is equal to

$$\log\left(L\left(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right)\right) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$$

To find the maximum of σ^2 and $\boldsymbol{\beta}$, we get the derivative of $\log\left(L\left(\widehat{\boldsymbol{\beta}}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right)\right)$ with regard to these parameters

$$\frac{\log\left(L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X})\right)}{\partial\boldsymbol{\beta}} = \frac{\left[\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)\boldsymbol{\beta} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}\right]}{\sigma^2}$$

$$\frac{\log\left(L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X})\right)}{\partial\sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$$

Now, by setting these derivatives equal to zero and solving the resulting equations for $\boldsymbol{\beta}$ and σ^2, we found that the estimates of these parameters are

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

$$\widehat{\sigma}^2 = \frac{1}{n}\left(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right)^{\mathrm{T}}\left(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right).$$

From this we can see that for each value of σ^2, the value of $\boldsymbol{\beta}$ that maximizes the likelihood is the same value that maximizes $-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$, which in turn

minimizes $(\boldsymbol{y}-\boldsymbol{X\beta})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X\beta})$, which is precisely the OLS of $\boldsymbol{\beta}, \widehat{\boldsymbol{\beta}}$. But when equating the derivative of $\log\left(L\left(\widehat{\boldsymbol{\beta}}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right)\right)$ to zero and solving for σ^2, the value of σ^2 that maximizes $L\left(\widehat{\boldsymbol{\beta}}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right)$ is $\widehat{\sigma}^2 = \frac{1}{n}\left(\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}\right)^{\mathrm{T}}\left(\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}\right)$.

Finally,

$$L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}) \leq L\left(\widehat{\boldsymbol{\beta}}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}\right) \leq L\left(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2; \boldsymbol{y}, \boldsymbol{X}\right)$$

and from here, the MLE of $\boldsymbol{\beta}$ and σ^2 are $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$, because it can be shown that the values of parameters that maximize the likelihood are unique when the design matrix $\boldsymbol{X}$ is of full column rank.

3.4 Fitting the Linear Multiple Regression Model via the Gradient Descent (GD) Method

The steepest descent method, also known as the gradient descent (GD) method, is a first-order iterative algorithm for minimizing a function (f). It is a central mechanism in statistical learning to training models (to estimate the parameters), for example, in neuronal networks and penalized regression models (Ridge and Lasso). It consists of successively updating the argument of the objective function in the direction of the steepest descent (along the negative of the gradient of the function), that is, in the direction in which f decreases most rapidly (Haykin 2009; Nocedal and Wright 2006). Specifically, each step of this algorithm is described by

$$\eta_{t+1} = \eta_t - \alpha \nabla f(\eta_t),$$

where $\nabla f(\eta_t)$ is the gradient vector of f evaluated in the current value η_t and α is a step size or learning rate parameter, which greatly determines the convergence behavior toward an optimal solution (Haykin 2009; Beysolow II 2017) and in neural networks it is popular for setting this at a small, fixed value (Warner and Misra 1996; Goodfellow et al. 2016). The learning rate parameter can be adaptative as well, that is, can be allowed to change at each step. For example, in the library Keras (see Chap. 11) that can be used for implementing and training neuronal networks models, there are several optimizers based on an adaptive gradient descendent algorithm such as Adam Adgrad, Adadelta, RMSprop, among others (Allaire and Chollet 2019). The ideal value of the step size would be the value that gives the larger reduction in each step, that is, the value of α that minimizes $f(\eta_t - \alpha \nabla f(\eta_t))$, which in general is difficult and expensive to obtain (Nocedal and Wright 2006).

Although the use of this algorithm could be avoided in an MLR, especially in small data sets, and also because of its slow convergence in linear systems (Burden and Faires 2011), here we will describe how this works when finding the optimal

beta coefficients in this model. First, the gradient of the residual sum of squares is given by

$$\nabla \mathrm{RSS}(\boldsymbol{\beta}) = 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right).$$

Then, the next update of beta coefficients in the gradient descent algorithm in this model is given by

$$\begin{aligned}\boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t - 2\alpha\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta}_t - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right)\\ &= \boldsymbol{\beta}_t - 2\alpha\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{\beta}_t - \boldsymbol{y})\\ &= \boldsymbol{\beta}_t + 2\alpha\boldsymbol{X}^{\mathrm{T}}\boldsymbol{e}_t,\end{aligned}$$

where $\boldsymbol{e}_t = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}_t$ is the vector of residuals that is obtained in the current iteration. One way to speed up the convergence of the algorithm is by choosing the ideal learning rate in each step, which, as was described before, is given by the value of α that minimizes $f(\eta_t - \alpha \nabla f(\eta_t))$, and in this case for the MLR model is given by (Nocedal and Wright 2006):

$$\alpha_t = \frac{\boldsymbol{e}_t^{\mathrm{T}}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{e}_t}{\boldsymbol{e}_t^{\mathrm{T}}\left(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\right)^2\boldsymbol{e}_t}.$$

Example 1 For numerical illustration, we considered a synthetic data set that consists of 100 observations and two covariates. The scatter plots in Fig. 3.1 show how the response variable (y) is related to the two covariates (x_1, x_2). By setting a value of 10^{-2} for the learning rate parameter, and as the stopping criterion a tolerance of 10^{-8} for the maximum norm of the difference between the current and next vector value, the beta coefficient obtained with the GD method is $\widehat{\boldsymbol{\beta}} = (5.0460764, 0.8551383, 2.1903356)$. For these synthetic examples, 12 iterations were necessary, while by changing the learning rate parameter to 10^{-3}, the number of iterations increased to 185, but we practically got the same results. Now, by using the "optimal" learning rate parameter described before for MLR with the same tolerance error (10^{-8}), the number of required iterations up to convergence is reduced to only 10 iterations. In general, the performance of the gradient descent depends greatly on the objective function and can be affected by the characteristics of the model, the dispersion of the data (explained variance of the predictors), and the dependence between the predictors, among others.

In the data set used in this example, the covariates are independent and the proportion of explained variances by the predictor is about 79% of the total variance of the response. By changing to a pair of moderately correlated covariates with correlation 0.75, while holding the same beta coefficient values, the variance of the residual (1.44), and the same sample size, we generate data where a greater proportion of variance is explained by the covariates (85.6%), but when applying the

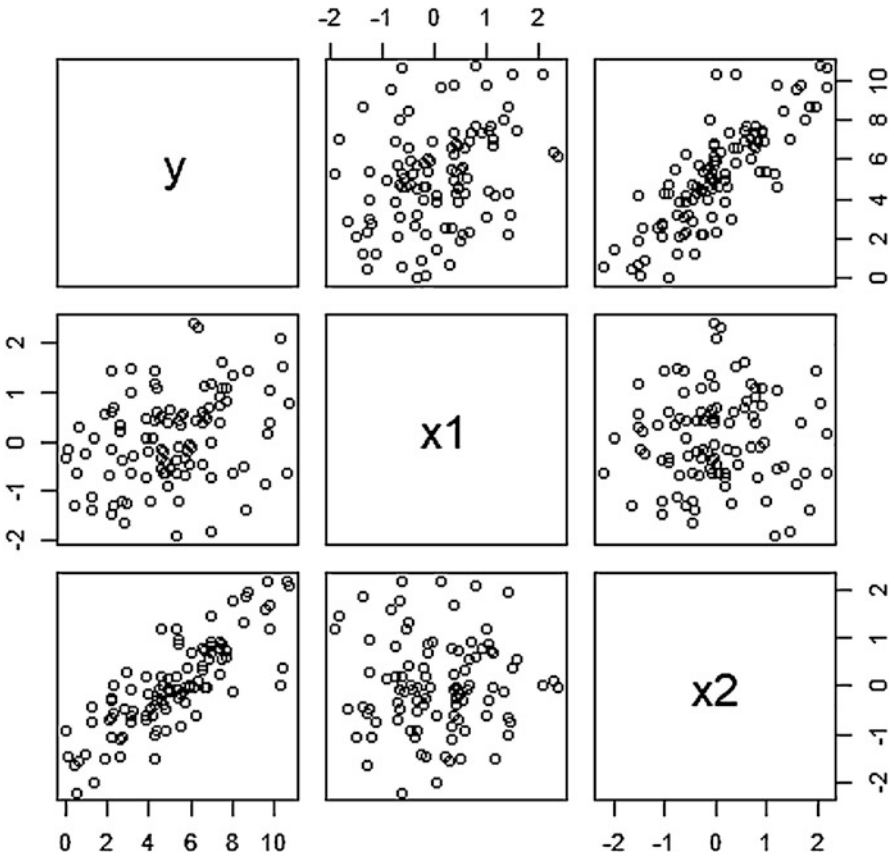

Fig. 3.1 Scatter plot of synthetic data generated from an MLR with two covariates

gradient descent with the same tolerance error (10^{-8}) as before, and learning rate values of 10^{-2} and 10^{-3}, the required number of iterations are about 5.75 times (69) and 3.5 times (649) the number required for the independent covariates case and the example described before, respectively.

Continuing with the last case of dependent variables, when using the optimal learning rate described before for the MLR, the number of iterations is reduced to 60, 9 less than when using the constant learning rate 10^{-2}.

By multiplying the beta coefficients used before by sqrt(0.1), the proportion of explained variance by the covariates is reduced to 27.30% and 37.2% in the same independent covariate (E3) and the same correlated covariate (E4) scenarios described before, respectively. With a tolerance error of 10^{-8} and with a learning rate equal to 10^{-2}, the required number of iterations are 183 and 66, for scenarios E3 and E4, respectively, while for a learning rate of 10^{-3} the required number of iterations are 617 and 1638. When using the "optimal" learning rate parameter, the required number of iterations is reduced to 17 and 56 for scenarios E3 and E4, respectively.

The R code used for implementing the GD method is given next.

```
##################R code for Example 1 ################################
rm(list=ls())
library(mvtnorm)
set.seed(1)
X = cbind(1,rmvnorm(100,c(0,0),diag(2)))
#Uncomment the next three lines code to simulate dependent covariables
#Sigma_X =0.75+0.25*diag(2)
#L = t(chol(Sigma_X))
#X = X%*%t(L)

betav = c(5,1,2.1)
#Uncomment the next line code to reduce the value of the beta cofficients
and reduce the proportion of variance of the response explained by the
features
betav = sqrt(0.1)*beta
y = X%*%betav + rnorm(100,0,1.2)
dat = data.frame(y=y,x1 = X[,2],x2=X[,3])
plot(dat)

alpha = 1e-2
#alpha = 1e-3
tol   = 1e-8
p = 2
betav_0 = c(mean(y),rep(0,p))
tol.e = 1
Iter = 0
tX = t(X)
XtX = X%*%t(X)
while(tol<tol.e)
{
 Iter = Iter + 1
 e = y-X%*%betav_0
 #Uncomment the next line code to use the optimal learning rate
 #alpha = (t(e)%*%XtX%*%e/(t(e)%*%(XtX)%*%XtX%*%e))[1,1]
 betav_t = betav_0 + alpha*tX%*%e
 tol.e = max(abs(betav_t-betav_0))
 betav_0 = betav_t
}
betav_t
tol.e
Iter
```

This code is only for illustrative purposes, that is, to illustrate in a very transparent way how the GD method can be implemented. Of course the existing statistical machine learning software programs implement this method and so there is no need to use this program for real applications, since the existing software programs that implement this method do a lot of work more efficiently and in a more user-friendly way.

3.5 Advantages and Disadvantages of Standard Linear Regression Models (OLS and MLR)

The MLR is a simple and computationally appealing class of models, but with many predictors (relative to the sample size) or nearly dependent features, it may result in large prediction error and/or large predictive intervals (Wakefield 2013). To appreciate the latter case (nearly dependent features), consider the spectral decomposition $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^{\mathrm{T}}$, where $\boldsymbol{\Lambda} = \mathrm{Diag}(\lambda_0, \ldots, \lambda_p)$ is a diagonal matrix with the eigenvalues of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ in decreasing order and $\boldsymbol{\Gamma}$ is an orthogonal matrix with columns corresponding to eigenvectors of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. Then the obtained variance–covariance matrix of the OLS estimator of $\widehat{\boldsymbol{\beta}}$ can be expressed as

$$\mathrm{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \sigma^2\left(\boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^{\mathrm{T}}\right)^{-1} = \sigma^2\boldsymbol{\Gamma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Gamma}^{\mathrm{T}}.$$

When the features are nearly dependent, some λ_j's will be "close" to zero and consequently the variance of some $\widehat{\beta}_j'$s will be high; this is even greater when the linear dependence of the features is strong (Wakefield 2013; Christensen 2011). This strong dependence between features is a problem of the OLS in MLR that is also reflected in the quality of the prediction performance, for example, when this is measured by the conditional expected prediction error (EPE) or mean squared error prediction that for an individual with feature $\boldsymbol{x}_o$ is given by

$$\begin{aligned}
\mathrm{EPE}(\boldsymbol{x}_o) &= E_{\boldsymbol{Y},Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(Y_o - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}\right)^2\right] \\
&= E_{\boldsymbol{Y},Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left\{\left[\left(Y_o - E(Y_0|\boldsymbol{x}_0) + E(Y_0|\boldsymbol{x}_0) - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}\right)\right]^2\right\} \\
&= E_{\boldsymbol{Y},Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(Y_o - E\left(Y_0|\boldsymbol{x}_0^{*\mathrm{T}}\right)\right)^2\right] + 2E_{Y_o|\boldsymbol{x}_o}\left[\left(Y_o - E\left(Y_0|\boldsymbol{x}_0^{*\mathrm{T}}\right)\right)\right]\left[E\left(Y_0|\boldsymbol{x}_0^{*\mathrm{T}}\right) - E_{Y|X}\left(\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}\right)\right] \\
&\quad + E_{\boldsymbol{Y},Y_o|\boldsymbol{X}}\left[\left(E\left(Y_0|\boldsymbol{x}_0^{*\mathrm{T}}\right) - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}\right)^2\right] \\
&= \sigma^2 + E_{\boldsymbol{Y},Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta} - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}\right)^2\right] = \sigma^2 + Var\left(\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}|\boldsymbol{x}_o\right) \\
&= \sigma^2 + \sigma^2\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\Gamma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{x}_o^* \\
&= \sigma^2\left(1 + \sum_{j=0}^{p}\frac{\left(x_{oj}^{**}\right)^2}{\lambda_j}\right),
\end{aligned}$$

where $\boldsymbol{x}_o^{**} = \boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{x}_o^* = \left(x_{o0}^{**}, \ldots, x_{op}^{**}\right)^{\mathrm{T}}$. This means that the average loss incurred (squared difference between the value to be predicted and the predicted value) by predicting Y_0 with its estimated mean under the MLR, $\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}$, is composed of intrinsic or irreducible data noise (first term) and the variance of $\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}$ (second term). The former cannot be avoided no matter how well the mean value of $Y_0 \mid x_0$, $E(Y_0|\, x_0)$, is

estimated, and the latter increases as the dependence of features is stronger. From this, it is apparent that the EPE is also affected by the strong dependence between features, which is a problem of the OLS in an MLR in a prediction context.

3.6 Regularized Linear Multiple Regression Model

3.6.1 Ridge Regression

Ridge regression, originally proposed as a method to combat multicollinearity, is also a common approach for controlling overfitting in an MLR model (Christensen 2011). It translates the OLS problem into the minimization of the penalized residual sum of squares defined as

$$\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta}) = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2,$$

where $\lambda \geq 0$ is known as the regularization or tuning parameter, which determines the level or degree to which the beta coefficients are shrunk toward zero. When $\lambda = 0$, the OLS is the solution to the beta coefficients, but when λ is large, the $\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta})$ is dominated by the penalization term, and the OLS solution has to shrink toward 0 (Christensen 2011). In general, when the number of parameters to be estimated is larger than the number of observations, the estimator can be highly variable. In this situation, the intuition of Ridge regression tries to alleviate this by constraining the sum of squares for the beta coefficients.

Note that $\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta})$ can be expressed as

$$\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta}) = \mathrm{RSS}(\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\beta},$$

where $\boldsymbol{D} =$ diag $(0, 1, \ldots, 1)$ is an identity matrix of dimension $(p + 1) \times (p + 1)$ but with one zero in its first entry. Then, the gradient of $\mathrm{RSS}_{\lambda}(\boldsymbol{\beta})$, that is, the first derivative with regard to $\boldsymbol{\beta}$ of $\mathrm{RSS}_{\lambda}(\boldsymbol{\beta})$, is

$$\nabla\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta}) = 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right) + 2\lambda\boldsymbol{D}\boldsymbol{\beta}.$$

Solving $\nabla\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta}) = \mathbf{0}$, the Ridge solution is given by

$$\widehat{\boldsymbol{\beta}}^{R}(\lambda) = \underset{\boldsymbol{\beta}}{\mathrm{argmin}}\mathrm{PRSS}_{\lambda}(\boldsymbol{\beta}) = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{D}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}.$$

This is a biased estimator of $\boldsymbol{\beta}$ because the conditional expected value is given by

$$E\left[\widehat{\boldsymbol{\beta}}^{R}(\lambda)\right] = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda \boldsymbol{D}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta}$$

but as will be described later, relative to the OLS estimator, by introducing a "small" bias, the variance or/and the EPE of this method could potentially be reduced (Wakefield 2013).

By using the method of Lagrange multipliers, the Ridge regression estimates of the $\boldsymbol{\beta}$ coefficients can be reformulated in a similar way to the OLS problem, but subject to the condition that the magnitude of the $\boldsymbol{\beta}_0 = (\beta_1, \ldots, \beta_p)^{\mathrm{T}}$ be less or equal to $t(\lambda)^{\frac{1}{2}}$, that is,

$$\widehat{\boldsymbol{\beta}}^{R}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \mathrm{RSS}(\boldsymbol{\beta})$$
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq t(\lambda),$$

where $t(\lambda)$ is a one-to-one function that produces an equivalent definition to the penalized OLS presentation of the Ridge regression described before (Wakefield 2013; Hastie et al. 2009, 2015). This constrained reformulation gives a more transparent role than the one played by the tuning parameter, and among other things, suggests a convenient and common way of redefining the Ridge estimator by standardizing the variables when these are of very different scales.

A graphic representation of this constraint problem for $\beta_0 = 0$ and $p = 2$ is given in Fig. 3.2, where the nested ellipsoids correspond to contour plots of RSS($\boldsymbol{\beta}$) and the green region is the restriction with $t(\lambda) = 3^2$, which contains the Ridge solution.

The MLR defined in (3.1) but now defined with the standardized variables is expressed as

$$\begin{aligned} y &= 1_n\mu + \boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s} + \epsilon \\ &= \boldsymbol{X}_s\boldsymbol{\beta}_s + \epsilon, \end{aligned}$$

where $\mathbf{1}_n$ is the column vector with 1's in all its entries, $\boldsymbol{X}_{1s} = \begin{bmatrix} x_{11s} & \cdots & x_{1ps} \\ \vdots & \vdots & \vdots \\ x_{n1s} & \cdots & x_{nps} \end{bmatrix}$, $x_{ijs} = \left(x_{ij} - \overline{x}_j\right)/s_j$, $s_j = \sqrt{\sum_{i=1}^{n}\left(x_{ij} - \overline{x}_j\right)^2/n}$, $j = 1, \ldots, p$; $\boldsymbol{X}_s = [\mathbf{1}_n\ \boldsymbol{X}_{1s}]$; $\boldsymbol{\beta}_s = \left(\mu,\ \boldsymbol{\beta}_{0s}^{\mathrm{T}}\right)^{\mathrm{T}}$; and $\boldsymbol{\beta}_{0s} = (\beta_{1s}, \ldots, \beta_{ps})^{\mathrm{T}}$.

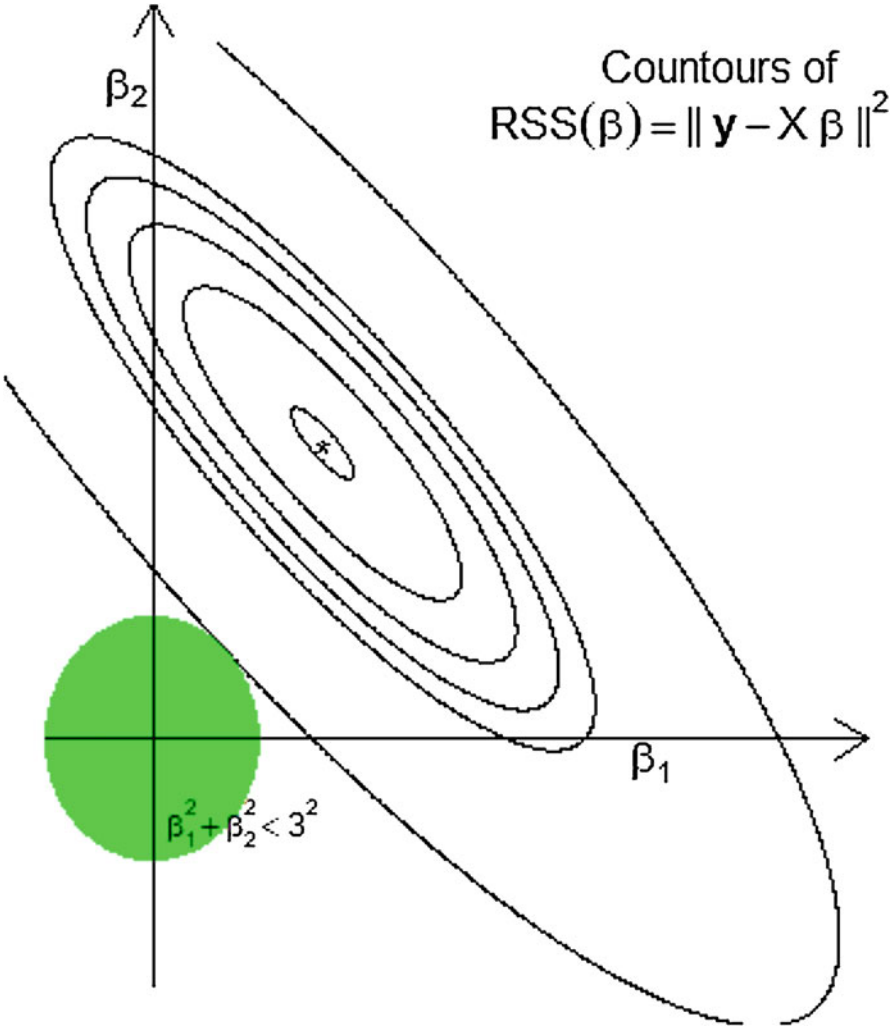

Fig. 3.2 Graphic representation of the Ridge solution of the OLS with restriction $\sum_{j=1}^{p} \beta_j^2 < 3^2$. The green region contains the Ridge solution for $t(\lambda) = 3^2$

Then, the redefined penalized residual sum squared under this model is

$$\begin{aligned} \text{PRSS}_\lambda(\boldsymbol{\beta}_s) &= \sum_{i=1}^{n} \left(y_i - \mu - \sum_{j=1}^{p} x_{ijs} \beta_{js} \right)^2 + \lambda \sum_{j=1}^{p} \beta_{js}^2 \\ &= \left(\boldsymbol{y} - \boldsymbol{X}_s^{\mathrm{T}} \boldsymbol{\beta}_s \right)^{\mathrm{T}} \left(\boldsymbol{y} - \boldsymbol{X}_s^{\mathrm{T}} \boldsymbol{\beta}_s \right) + \lambda \boldsymbol{\beta}_{0s}{}^{\mathrm{T}} \boldsymbol{D} \boldsymbol{\beta}_{0s}. \end{aligned}$$

The Ridge solution under this redefinition is like the one given before, but now

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_s^R(\lambda) &= \left(\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s + \boldsymbol{\lambda D}\right)^{-1}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{y} \\ &= \left(\begin{bmatrix} 1_n^{\mathrm{T}} \\ \boldsymbol{X}_{1s}^{\mathrm{T}} \end{bmatrix}[1_n\ \boldsymbol{X}_{1s}] + \lambda \boldsymbol{D}\right)^{-1}\begin{bmatrix} 1_n^{\mathrm{T}} \\ \boldsymbol{X}_{1s}^{\mathrm{T}} \end{bmatrix}\boldsymbol{y} \\ &= \begin{bmatrix} n & 0_n^{\mathrm{T}} \\ 0_n & \boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I_p} \end{bmatrix}^{-1}\begin{bmatrix} 1_n^{\mathrm{T}} y \\ \boldsymbol{X}_{1s}^{\mathrm{T}} y \end{bmatrix} \\ &= \begin{bmatrix} \overline{y}_n \\ \widehat{\boldsymbol{\beta}}_{0s}(\lambda) \end{bmatrix},\end{aligned}$$

where $\overline{y}_n = \sum_{i=1}^{n} y_i/n$ is the sample mean of the responses and $\widehat{\boldsymbol{\beta}}_{0s}(\lambda) = \left(\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I_p}\right)^{-1}\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{y}$ is the Ridge estimator of $\boldsymbol{\beta}_{0s}$. The mean value of this Ridge solution is

$$\mathrm{E}\left[\widehat{\boldsymbol{\beta}}_s^R(\lambda)\right] = \begin{bmatrix} \mu \\ E\left[\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right] \end{bmatrix},$$

where $E\left[\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right] = \left(\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I_p}\right)^{-1}\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s}$ is the expected value of the Ridge estimator of $\boldsymbol{\beta}_{0s}$. The variance–covariance matrix is

$$\begin{aligned}\mathrm{Var}\left(\widehat{\boldsymbol{\beta}}_s^R(\lambda)\right) &= \left(\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s + \boldsymbol{\lambda D}\right)^{-1}\boldsymbol{X}_s^{\mathrm{T}}\mathrm{Var}(\boldsymbol{y})\boldsymbol{X}_s\left(\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s + \boldsymbol{\lambda D}\right)^{-1} \\ &= \sigma^2\begin{bmatrix} n & 0_n^{\mathrm{T}} \\ 0_n & \boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I_p} \end{bmatrix}^{-1}\begin{bmatrix} 1_n^{\mathrm{T}} \\ \boldsymbol{X}_{1s}^{\mathrm{T}} \end{bmatrix}[1_n\ \boldsymbol{X}_{1s}]\begin{bmatrix} n & 0_n^{\mathrm{T}} \\ 0_n & \boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I_p} \end{bmatrix}^{-1} \\ &= \sigma^2\begin{bmatrix} 1/n & 0_n^{\mathrm{T}} \\ 0_n & \mathrm{Var}\left(\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right) \end{bmatrix},\end{aligned}$$

where $\mathrm{Var}\left(\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right) = \left(\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I}_p\right)^{-1}\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s}\left(\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} + \lambda \boldsymbol{I}_p\right)^{-1}$. So, because in this standardized way, the Ridge solution of the intercept (μ) is the sample mean of the observed responses, and the correlation of this with the rest of the estimated parameters ($\widehat{\boldsymbol{\beta}}_{0s}(\lambda)$) is null, in the literature it is common to handle this parameter separately from all other coefficients ($\boldsymbol{\beta}_{0s}$) (Christensen 2011).

Note that

$$E\left[\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right] = \boldsymbol{\Gamma}_s\left(\boldsymbol{\Lambda}_s + \lambda \boldsymbol{I_p}\right)^{-1}\boldsymbol{\Lambda}_s\boldsymbol{\beta}_{0s}^*$$

and

$$\mathrm{Var}\left(\widehat{\boldsymbol{\beta}}_{0s}(\lambda)\right) = \boldsymbol{\Gamma}_s\left(\boldsymbol{\Lambda}_p + \lambda \boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Lambda}_s\left(\boldsymbol{\Lambda}_p + \lambda \boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Gamma}_s^{\mathrm{T}},$$

where $\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} = \boldsymbol{\Gamma}_s\boldsymbol{\Lambda}_s\boldsymbol{\Gamma}_s^{\mathrm{T}}$ is the spectral decomposition of $\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s}$ and $\boldsymbol{\beta}_{0s}^{*} = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s}$. So the conditional expected prediction error at $\boldsymbol{x}_o$ when using the Ridge solution is

$$\begin{aligned}
\mathrm{EPE}_{\lambda}(\boldsymbol{x}_o) &= E_{Y,Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(Y_o - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)\right)^2\right] \\
&= E_{Y,Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(Y_o - E(Y_0|\boldsymbol{x}_0) + E(Y_0|\boldsymbol{x}_0) - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)\right)^2\right] \\
&= \sigma^2 + E_{Y,Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta}_s - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)\right)^2\right] \\
&= \sigma^2 + \left[\left(\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta}_s - \boldsymbol{x}_o^{*\mathrm{T}}E_{Y|X}\left(\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)\right)\right)^2\right] + \mathrm{Var}\left(\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)|\boldsymbol{x}_{\boldsymbol{o}}\right) \\
&= \sigma^2 + \left[\left(\mu + \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\beta}_{0s} - \mu - \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_s\left(\boldsymbol{\Lambda}_s + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Lambda}_s\boldsymbol{\beta}_{0s}^{*}\right)^2\right] + \sigma^2\left[\frac{1}{n} + \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_s\left(\boldsymbol{\Lambda}_p + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Lambda}_s\left(\boldsymbol{\Lambda}_p + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Gamma}_s^{T}\boldsymbol{x}_o\right] \\
&= \sigma^2 + \left[\left(\boldsymbol{x}_o^{**\mathrm{T}}\boldsymbol{\beta}_{0s}^{*} - \boldsymbol{x}_o^{**\mathrm{T}}\left(\boldsymbol{\Lambda}_s + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Lambda}_s\boldsymbol{\beta}_{0s}^{*}\right)^2\right] + \sigma^2\left[\frac{1}{n} + \boldsymbol{x}_o^{**\mathrm{T}}\left(\boldsymbol{\Lambda}_p + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{\Lambda}_s\left(\boldsymbol{\Lambda}_p + \lambda\boldsymbol{I}_{\boldsymbol{p}}\right)^{-1}\boldsymbol{x}_o^{**}\right] \\
&= \sigma^2 + \left[\sum_{j=1}^{p}\left(1 - \frac{\lambda_j}{\lambda_j + \lambda}\right)x_{oj}^{**}\beta_{js}^{*}\right]^2 + \sigma^2\left(\frac{1}{n} + \sum_{j=1}^{p}\frac{\lambda_j}{\left(\lambda_j + \lambda\right)^2}\left(x_{oj}^{**}\right)^2\right),
\end{aligned}$$

where $\boldsymbol{x}_o^{**} = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{x}_o^{*} = \left(x_{o0}^{**}, \ldots, x_{op}^{**}\right)^{\mathrm{T}}$ and $\boldsymbol{\beta}_{0s}^{*} = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} = \left(\beta_{1s}^{*}, \ldots, \beta_{ps}^{*}\right)^{\mathrm{T}}$. The second and third terms of the last equality correspond to the squared bias and the variance of $\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^{R}(\lambda)$ as an estimator of $\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta}_s$, respectively. By setting $\lambda = 0$, this EPE corresponds to the EPE of the OLS prediction but with standardized variables, while by letting λ be very large, the variance will decrease and the squared bias will increase.

More importantly, because the derivative of $\mathrm{EPE}_{\lambda}(\boldsymbol{x}_o)$ with respect to λ, $\frac{d}{d\lambda}\mathrm{PE}_{\lambda}(\boldsymbol{x}_o)$, is a right continuous function at $\lambda = 0$, and for $\boldsymbol{X}_{1s}$ of full column rank, $\lim_{\lambda\to 0^{+}}\frac{d}{d\lambda}\mathrm{PE}_{\lambda}(\boldsymbol{x}_o) = -2\sigma^2\sum_{j=1}^{p}\frac{\left(x_{oj}^{**}\right)^2}{\lambda_j^2} = c$; then for $\epsilon = -\frac{c}{2} > 0$, we have that $\lambda^{*} > 0$ such that $\left|\frac{d}{d\lambda}\mathrm{PE}_{\lambda}(\boldsymbol{x}_o) - c\right| < \epsilon$ for $\lambda < \lambda^{*}$. From this we have that $\frac{d}{d\lambda}\mathrm{PE}_{\lambda}(\boldsymbol{x}_o) < -\frac{c}{2} + c = \frac{c}{2} < 0$ for all $\lambda < \lambda^{*}$, for some $\lambda^{*} > 0$. Then, at least in the interval $[0, \lambda^{*}]$, the expected prediction error at $\boldsymbol{x}_o$ shows a decreasing behavior, which indicates that there is a value of λ such that with the Ridge regression estimation of beta coefficients, we can get a smaller prediction error than with the OLS prediction. Figure 3.3 shows a graphic representation of this behavior of Ridge prediction, where the lower EPE is reached at about $\lambda =$ exp (2.22). Figure 3.3 also shows the increasing and decreasing behavior of the bias-squared and the variance involved.

When $\boldsymbol{X}_{1s}$ is not full column rank, the previous argument regarding the behavior of the EPE of the Ridge solution is already not valid directly, but it could be used for

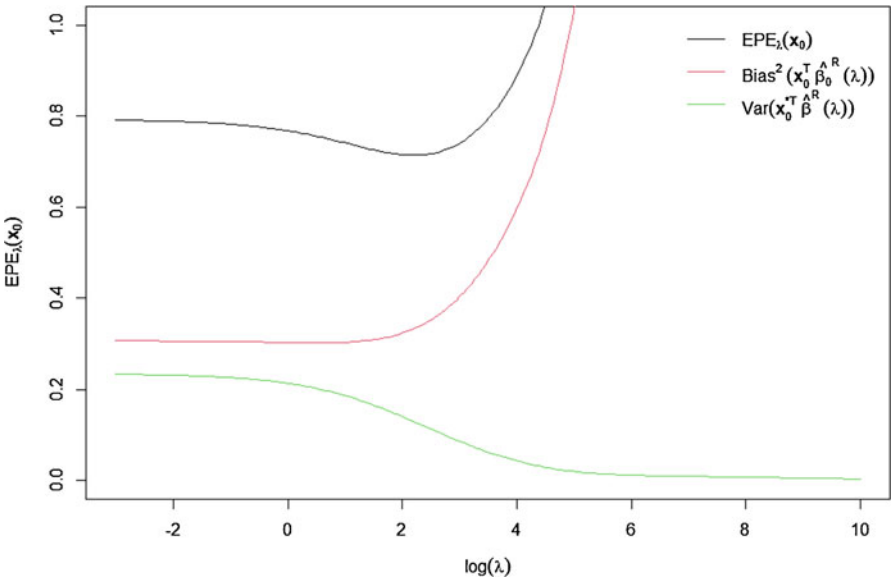

Fig. 3.3 Behavior of the expected prediction error at $\boldsymbol{x}_o$ of the Ridge solution

validating part of the more general case. To see this, first note that the spectral decomposition of $\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s}$ can be reduced to

$$\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s} = [\boldsymbol{\Gamma}_{1s}\ \boldsymbol{\Gamma}_{2s}]\begin{bmatrix}\boldsymbol{\Lambda}_{1s} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \boldsymbol{\Lambda}_{2s}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Gamma}_{1s}^{\mathrm{T}} \\ \boldsymbol{\Gamma}_{2s}^{\mathrm{T}}\end{bmatrix} = \boldsymbol{\Gamma}_{1s}^{\mathrm{T}}\boldsymbol{\Lambda}_{1s}\boldsymbol{\Gamma}_{1s},$$

where $\boldsymbol{\Lambda}_{1s} = \mathrm{Diag}\left(\lambda_1, \ldots, \lambda_{p^*}\right)$, $p^* = \mathrm{rank}\,(\mathbf{X}_{1c})$ is the rank of design matrix and $\boldsymbol{\Lambda}_{2s}$ is the null matrix of order $(p - p^*) \times (p - p^*)$. Furthermore, because $\boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\Gamma}_s = \boldsymbol{I}_p$ implies that $\boldsymbol{\Gamma}_{2s}^{\mathrm{T}}\boldsymbol{X}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1s}\boldsymbol{\Gamma}_{2s} = \mathbf{0}$, which in turn implies that $\boldsymbol{X}_{1s}\boldsymbol{\Gamma}_{2s} = \mathbf{0}$, then the MLR can be conveniently expressed by

$$\begin{aligned}\boldsymbol{y} &= 1_n\mu + \boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s} + \epsilon \\ &= 1_n\mu + \boldsymbol{X}_{1s}\boldsymbol{\Gamma}_s\boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} + \epsilon \\ &= 1_n\mu + \boldsymbol{X}_{1s}[\boldsymbol{\Gamma}_{1s}\ \boldsymbol{\Gamma}_{2s}]\boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} + \epsilon \\ &= 1_n\mu + [\boldsymbol{X}_{1s}\boldsymbol{\Gamma}_{1s}\ \boldsymbol{X}_{1s}\boldsymbol{\Gamma}_{2s}]\boldsymbol{\beta}_{0s}^* + \epsilon \\ &= 1_n\mu + \boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{01s}^* + \epsilon,\end{aligned}$$

where $\boldsymbol{\beta}_{0s}^* = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s}$, $\boldsymbol{X}_{1s}^* = \boldsymbol{X}_{1s}\boldsymbol{\Gamma}_{1s}$, and $\boldsymbol{\beta}_{0s}^* = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} = \left[\boldsymbol{\beta}_{0s}^T\boldsymbol{\Gamma}_{1s}, \boldsymbol{\beta}_{0s}^{\mathrm{T}}\boldsymbol{\Gamma}_{2s}\right]^{\mathrm{T}} = \left[\boldsymbol{\beta}_{01s}^{*\,\mathrm{T}}\ \boldsymbol{\beta}_{02s}^{*\mathrm{T}}\right]^{\mathrm{T}}$.

Also, from similar arguments, note that the penalized residual sum of squares of the Ridge solution can be expressed by

$$
\begin{aligned}
&(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s})^{\mathrm{T}}(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s}) + \lambda\boldsymbol{\beta}_{0s}^{\mathrm{T}}\boldsymbol{\beta}_{0s} \\
&= \left(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{01s}^*\right)^{\mathrm{T}}\left(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{01s}^*\right) + \lambda\boldsymbol{\beta}_{0s}^{\mathrm{T}}\left(\boldsymbol{\Gamma}_{1s}^{\mathrm{T}}\boldsymbol{\Gamma}_{1s} + \boldsymbol{\Gamma}_{2s}^{\mathrm{T}}\boldsymbol{\Gamma}_{2s}\right)\boldsymbol{\beta}_{0s} \\
&= \left(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{01s}^*\right)^{\mathrm{T}}\left(\boldsymbol{y} - 1_n\mu - \boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{01s}^*\right) + \lambda\boldsymbol{\beta}_{01s}^{*\mathrm{T}}\boldsymbol{\beta}_{01s}^* + \lambda\boldsymbol{\beta}_{02s}^{*\mathrm{T}}\boldsymbol{\beta}_{02s}^*
\end{aligned}
$$

This function of $\boldsymbol{\beta}_{0s}^*$ is minimized at $\widetilde{\boldsymbol{\beta}}_{0s}^*(\lambda) = \left(\widetilde{\boldsymbol{\beta}}_{01s}^{*\mathrm{T}}(\lambda), \boldsymbol{0}_{p-p^*}^{\mathrm{T}}\right)^{\mathrm{T}}$, where $\widetilde{\boldsymbol{\beta}}_{01s}^{*\mathrm{T}}(\lambda) = \left(\boldsymbol{\Lambda}_{1s} + \lambda\boldsymbol{I}_{p^*}\right)^{-1}\boldsymbol{\Gamma}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1S}^{\mathrm{T}}\boldsymbol{y}$ is the Ridge solution of the MLR expressed in terms of $\boldsymbol{X}_{1s}^*\boldsymbol{\beta}_{0s}^*$. Furthermore, because $\boldsymbol{\beta}_{0s}^* = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s}$ is a non-singular transformation, the original Ridge solution of $\boldsymbol{\beta}_s$ can be expressed in terms of $\widetilde{\boldsymbol{\beta}}_{0s}^*(\lambda)$ as $\widehat{\boldsymbol{\beta}}_s(\lambda) = \left(\overline{y}_n, \widehat{\boldsymbol{\beta}}_{0s}^{*\mathrm{T}}(\lambda)\right)^{\mathrm{T}}$, where $\widehat{\boldsymbol{\beta}}_{0s}(\lambda) = \boldsymbol{\Gamma}_s\widehat{\boldsymbol{\beta}}_{0s}^{*\mathrm{T}}(\lambda) = \boldsymbol{\Gamma}_{1s}\widetilde{\boldsymbol{\beta}}_{01s}^*(\lambda)$. Then, in a similar fashion as before, the conditional expected prediction error at $\boldsymbol{x}_o$ by using the Ridge solution in this case can be computed as

$$
\begin{aligned}
&\mathrm{EPE}_\lambda(\boldsymbol{x}_o) \\
&= \sigma^2 + E_{\boldsymbol{Y},Y_o|\boldsymbol{X},\boldsymbol{x}_o}\left[\left(\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta}_s - \boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^R(\lambda)\right)^2\right] \\
&= \sigma^2 + \left[\left(\boldsymbol{x}_o^{*\mathrm{T}}\boldsymbol{\beta}_s - \boldsymbol{x}_o^{*\mathrm{T}}E_{Y|X}\left(\widehat{\boldsymbol{\beta}}_s^R(\lambda)\right)\right)^2\right] + \mathrm{Var}\left(\boldsymbol{x}_o^{*\mathrm{T}}\widehat{\boldsymbol{\beta}}_s^R(\lambda)|\boldsymbol{x}_o\right) \\
&= \sigma^2 + \left[\left(\mu + \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_s\boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} - \mu - \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_{1s}E_{Y|X}\left(\widehat{\boldsymbol{\beta}}_{0s}^{*\mathrm{T}}(\lambda)\right)\right)^2\right] + \left[\frac{\sigma^2}{n} + \mathrm{Var}\left(\boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_s\widehat{\boldsymbol{\beta}}_{0s}^{*\mathrm{T}}(\lambda)|\boldsymbol{x}_o\right)\right] \\
&= \sigma^2 + \left[\left(\mu + \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_s\boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{0s} - \mu - \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_{1s}\left(\boldsymbol{\Lambda}_{1s} + \lambda\boldsymbol{I}_{p^*}\right)^{-1}\boldsymbol{\Gamma}_{1s}^{\mathrm{T}}\boldsymbol{X}_{1c}^{\mathrm{T}}\boldsymbol{X}_{1c}\boldsymbol{\beta}_{0S}\right)^2\right] \\
&\quad + \sigma^2\left[\frac{1}{n} + \boldsymbol{x}_o^{\mathrm{T}}\boldsymbol{\Gamma}_{1s}\left(\boldsymbol{\Lambda}_{1s} + \lambda\boldsymbol{I}_{p^*}\right)^{-1}\boldsymbol{\Lambda}_{1s}\left(\boldsymbol{\Lambda}_{1s} + \lambda\boldsymbol{I}_{p^*}\right)^{-1}\boldsymbol{\Gamma}_{1s}^{T}\boldsymbol{x}_o\right] \\
&= \begin{cases}
\sigma^2 + \left[\left(\sum_{j=1}^{p^*}\left(1 - \frac{\lambda_j}{\lambda_j + \lambda}\right)x_{oj}^*\beta_{oj}^*\right)^2\right] + \sigma^2\left[\frac{1}{n} + \sum_{j=1}^{p^*}\frac{\lambda_j}{\left(\lambda_j + \lambda\right)^2}\left(x_{oj}^*\right)^2\right] & \text{if } x_o = \boldsymbol{\Gamma}_{1s}\boldsymbol{a}_1 \\
\sigma^2 + \left[\boldsymbol{x}_o^T\boldsymbol{\beta}_{0s}\right]^2 + \frac{\sigma^2}{n} & \text{if } x_o = \boldsymbol{\Gamma}_{2s}\boldsymbol{a}_2,
\end{cases}
\end{aligned}
$$

where $\boldsymbol{x}_o^* = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{x}_o = \left[x_{o1}^*, \ldots, x_{op}^*\right]^{\mathrm{T}}$, $\boldsymbol{\beta}_o^* = \boldsymbol{\Gamma}_s^{\mathrm{T}}\boldsymbol{\beta}_{os} = \left[\beta_{o1}^*, \ldots, \beta_{op}^*\right]^{\mathrm{T}}$, and $\boldsymbol{a}_1 \in \mathbb{R}^{p^*}$ and $\boldsymbol{a}_2 \in \mathbb{R}^{p-p^*}$. So, using a similar argument as before, in the first case ($x_o = \boldsymbol{\Gamma}_{1s}\boldsymbol{a}_1$), the value of $\lambda > 0$ is such that the expected prediction error at $\boldsymbol{x}_o$ is better than that obtained with the OLS approach, $\widehat{\boldsymbol{\beta}}_s(0) = \lim_{\lambda\to 0}\widehat{\boldsymbol{\beta}}_s(\lambda) = \left[\overline{y}_n, \left(\boldsymbol{\Gamma}_{1s}\Lambda_{1s}^{-1}\boldsymbol{\Gamma}_{1s}^{\mathrm{T}}X_{1c}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\right]^{\mathrm{T}}$. In the second case, $x_o = \boldsymbol{\Gamma}_{2s}\boldsymbol{a}_2$, the EPE($\boldsymbol{x}_o$) in both approaches is the same and doesn't depend on λ, so in such cases, no improved gain with regard to the Ridge solution is achieved. A third case was included, that is, when the target feature is of the form $\boldsymbol{x}_o = \boldsymbol{\Gamma}_{1s}\boldsymbol{a}_1 + \boldsymbol{\Gamma}_{2s}\boldsymbol{a}_2$. In this case, under the described argument, the advantage of Ridge regression over the OLS approach in a prediction context is not clear.

However, in practice, we don't know the true value of the parameters, and we need to evaluate the test error in all possible values of the training sample, which we also don't have. So a common way to choose the λ value is by cross-validation. For more details about validation strategies, see Chap. 4. For example, with a k-fold CV, the complete data set is divided into K balanced disjoint subsets, S_k, $k = 1, \ldots, K$. One subset is used as validation and the rest are used to fit the model in each value of a chosen grid of values of λ. This procedure is repeated K times, where each time a subset in the partition is taken as the validation set. A more detailed k-fold CV procedure is described below:

1. First, choose a grid of values of λ, $\lambda = (\lambda_1, \ldots, \lambda_L)$.
2. Remove the subset S_k and for each value λ_l in the grid, fit the model with the remaining $K - 1$ elements of the partition denoted by $\widehat{\boldsymbol{\beta}}^R_{-k}(\lambda_l)$, the corresponding Ridge estimation of $\boldsymbol{\beta}$, and compute the average prediction error across all observations in the validation set S_k as

$$\widehat{\text{APE}}_{-k}(\lambda_l) = \frac{1}{|S_k|} \sum_{y_i \in S_k} \left(y_i - \boldsymbol{x}_i^{*\text{T}} \widehat{\boldsymbol{\beta}}^R_{-k}(\lambda_l) \right)^2,$$

   where $|S_k|$ denotes the total observations in partition k.
3. Choose as the best value of λ in the grid $\left(\widetilde{\lambda}^*\right)$, the one with the lower average prediction error across all partitions, that is

$$\widetilde{\lambda}^* = \arg\min_{\lambda_l} \widehat{\text{APE}}(\lambda_l),$$

   where $\widehat{\text{APE}}(\lambda_l) = \frac{1}{K} \sum_{k=1}^{K} \widehat{\text{APE}}_{-k}(\lambda_l)$.
4. Once $\widetilde{\lambda}^*$ is chosen, we fit the model with the complete data set and the prediction of new individuals with feature $\boldsymbol{x}_o$ can be made with $\widehat{y}_i = \boldsymbol{x}_o^{*T} \widehat{\boldsymbol{\beta}}^R \left(\widetilde{\lambda}^*\right)$, where $\widehat{\boldsymbol{\beta}}^R \left(\widetilde{\lambda}^*\right)$ is the Ridge estimation of $\boldsymbol{\beta}$ at $\lambda = \widetilde{\lambda}^*$.

It is important to point out that very often the performance of the model needs to be evaluated for comparison purposes with other competing models. A common way to do this is to split the data set several times into two subsets, one for training the model (D_{tr}) (to fit the model) and the other for testing (D_{tst}) it, in which the predictive ability of a model is tested. In each splitting, only the training data set (D_{tr}) is used to train the model (by steps 1–3 before fitting the whole training data set), and the prediction evaluation of the fitted model is made with the testing data set, as explained before in point 4. The prediction evaluation of the testing data set is done by an empirical "estimate" of the EPE, $\text{MSE} = \frac{1}{|D_{\text{tst}}|} \sum_{i \in D_{\text{tst}}} \left(y_i - \boldsymbol{x}_o^{*\text{T}} \widehat{\boldsymbol{\beta}}^R \left(\widetilde{\lambda}^*\right), \right)^2$, and

finally, an average evaluation of the performance of the model is obtained across all chosen splittings. See Chap. 4 for more explicit details.

The Ridge solution can also be obtained from a Bayesian formulation. To do this, consider the MLR model described before with standardized features and the vector of residuals distributed as $N_n(\mathbf{0}, \sigma^2 \boldsymbol{I}_n)$. With this assumption, the vector of responses $\boldsymbol{y}$ is distributed in a multivariate normal distribution with vector mean $\mathbf{1}_n\mu + \boldsymbol{X}_{1s}\boldsymbol{\beta}_{0s}$ and variance–covariance matrix $\sigma^2\boldsymbol{I}_n$. Then, to complete the Bayesian formulation, assume $\boldsymbol{\beta}_{os} \sim N_p\left(\mathbf{0}, \sigma_\beta^2 \boldsymbol{I}_p\right)$ as the prior distribution of the beta coefficients in $\boldsymbol{\beta}_{0s}$ and a "flat" prior for the intercept μ, where σ^2 and σ_β^2 are known. Under this Bayesian specification, the posterior distribution of $\boldsymbol{\beta}_s$ is

$$
\begin{aligned}
& f(\boldsymbol{\beta}_s|\boldsymbol{y}, \boldsymbol{X}_{1s}) \\
& \propto \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}_s\boldsymbol{\beta}_s)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}_s\boldsymbol{\beta}_s)\right] \exp\left(-\frac{1}{2\sigma_\beta^2}\boldsymbol{\beta}_{0s}^{\mathrm{T}}\boldsymbol{\beta}_{0s}\right) \\
& \propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\beta}_s^{\mathrm{T}}\left(\sigma_\beta^{-2}\boldsymbol{D} + \sigma^{-2}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s\right)\boldsymbol{\beta}_s - 2\sigma^{-2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{X}_s\boldsymbol{\beta}_s\right]\right\} \\
& \propto \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}_s - \widetilde{\boldsymbol{\beta}}_s\right)^{\mathrm{T}}\widetilde{\boldsymbol{\Sigma}}_\beta^{-1}\left(\boldsymbol{\beta}_s - \widetilde{\boldsymbol{\beta}}_s\right)\right\},
\end{aligned}
$$

where $\widetilde{\boldsymbol{\Sigma}}_\beta = \left(\sigma_\beta^{-2}\boldsymbol{D} + \sigma^{-2}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s\right)^{-1} = \sigma^2\left(\sigma^2/\sigma_\beta^2\boldsymbol{D} + \boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s\right)^{-1}$, $\widetilde{\boldsymbol{\beta}}_s = \sigma^{-2}\widetilde{\boldsymbol{\Sigma}}_\beta\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{y}$, and $\boldsymbol{D}$ is the diagonal penalty matrix. That is, the posterior distribution of $\boldsymbol{\beta}_s$ is a multivariate normal distribution with vector mean $\widetilde{\boldsymbol{\beta}}_s = \sigma^{-2}\widetilde{\boldsymbol{\Sigma}}_\beta\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{y} = \left(\sigma^2/\sigma_\beta^2\boldsymbol{D} + \boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s\right)^{-1}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{y}$ and variance–covariance matrix $\widetilde{\boldsymbol{\Sigma}}_\beta = \sigma^2\left(\sigma^2/\sigma_\beta^2\boldsymbol{D} + \boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{X}_s\right)^{-1}$. Then, by taking $\lambda = \sigma^2/\sigma_\beta^2$, we have that the mean/mode of the posterior distribution of $\boldsymbol{\beta}_s$ coincides with the Ridge estimation described before, $\widetilde{\boldsymbol{\beta}}^R(\lambda)$.

Example 2 We considered a genomic example to illustrate the Ridge regression approach and the CV process to choose the learning parameter λ (WheatMadaToy, PH the response). This data set consists of 50 observations corresponding to 50 lines and a relationship genomic matrix computed from marker information. Table 3.1 shows the prediction behavior of the Ridge and the OLS approaches in terms of the MSEP, across five different splittings obtained by partitioning the complete data set into five subsets: the data of a subset are used as a testing set and the rest to train the model. For training the model, a five-fold cross-validation (5FCV) was used along the lines following steps 1–3 described before, and the prediction performance was done following step 4.

Table 3.1 Prediction behavior of the Ridge and OLS regression models across different partitions of the complete data set: one subset of the partition (20%) is used for evaluating the performance of the model and the rest (80%) for training the model. RR denotes Ridge regression method

Partition	MSE RR	MSE OLS
1	325.40	379.33
2	433.19	454.37
3	803.76	1341.35
4	319.53	312.81
5	403.62	555.10
Average	457.10	608.59

Table 3.1 indicates that in four out of five partitions, the Ridge regression shows less MSE than the corresponding OLS approach. In all these cases, the MSE of the OLS was, on average, 31.46% greater than that of the Ridge regression approach, and in general, on average, by 31.14% (MSE = 421.8834 for Ridge and MSE = 655.8596 for OLS). From this, we have that the Ridge regression approach shows a better prediction performance than the OLS. The large variation of the MSE between folds observed in this example could indicate that for obtaining a more precise comparison between models, a larger number of partitions need to be used. Often, the use of more partitions is avoided when larger data sets are used in applications.

The R code used for obtaining this result is the following:

```
######################R code for Example 2 ############################
rm(list=ls())
library(BMTME)
data("WheatMadaToy")
dat_F = phenoMada
dim(dat_F)
dat_F$GID = as.character(dat_F$GID)
G = genoMada
eig_G = eigen(G)
G_0.5 = eig_G$vectors%*%diag(sqrt(eig_G$values))%*%t(eig_G$vectors)
X = G_0.5
y = dat_F$PH
n = length(y)
source('TR_RR.R')
#5FCV
set.seed(3)
K = 5
Tab = data.frame()
Grpv = findInterval(cut(sample(1:n,n),breaks=K),1:n)
for(i in 1:K)
{
 Pos_tr = which(Grpv!=i)
 y_tr = y[Pos_tr]
 X_tr = X[Pos_tr,]
 TR_RR = Tr_RR_f(y_tr,X_tr,K=5,KG=100,KR=1)
 lambv = TR_RR$lambv
```

```
 #Tst
 y_tst = y[-Pos_tr]; X_tst = X[-Pos_tr,]
 #RR
 Pred_RR = Pred_RR_f(y_tst,X_tst,TR_RR)
 #OLS
 Pred_ols = Pred_ols_f(y_tst,X_tst,y_tr,X_tr)
 Tab = rbind(Tab,data.frame(Sim=i,MSEP_RR = Pred_RR$MSEP,
                MSEP_ols = Pred_ols$MSEP))
 cat('i = ', i,'\n')
}
Tab
```

Tr_RR_f, Pred_RR_f, and Pred_ols_f are R functions accessed by the command source('TR_RR.R'), where TR_RR.R is the file R script defined in Appendix 1.

From the last code, three things are important to point out:

1. The Grpv contains the information of the K=5 folds for the outer CV implemented and each time the model is trained with $K - 1$ and tested with the remaining fold.
2. The function that trains the model under Ridge regression is called Tr_RR_f, while the function that obtains the predictions of the testing set of this trained model is Pred_RR_f; both functions are fully described in Appendix 1.
3. The predictions under the OLS method are obtained with the function Pred_ols_f, which is also fully detailed in Appendix 1. It is important to point out that the function Tr_RR_f internally implements an inner k-fold CV to tune the hyperparameter λ required in Ridge regression.

Example 3 (Simulation) To get a better idea about the behavior of the Ridge solution, here we report the results of a small simulation study in a scenario where the number of observations (n=100) is less than the number of features ($p = 500$) and these are moderately correlated. Specifically, we generated 100 data sets, each of size 100, from the following model:

$$y_i = 5 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0 + \epsilon_i,$$

where the vector of beta coefficients ($\boldsymbol{\beta}_0$) was set to the values shown in Fig. 3.4, and the features of all the individuals in each data set were generated from a multivariate normal distribution centered on the null vector and variance–covariance matrix $\boldsymbol{\Sigma} = 0.25\boldsymbol{I}_p + 0.75\boldsymbol{J}_p$, where $\boldsymbol{I_p}$ and $\boldsymbol{J}_p$ are the identity matrix and matrix of ones of dimension $p \times p$. The random errors (ϵ_i) were simulated from a normal distribution with mean 0 and variance 0.025.

The behavior of the Ridge and OLS solutions across the 100 simulated data sets is shown in Fig. 3.5. The MSE of Ridge regression is located on the x-axis and the corresponding MSE of the OLS is located on the y-axis. On average, the OLS resulted in an MSE equal to 808.81, which is 30.59% larger than the average MSE

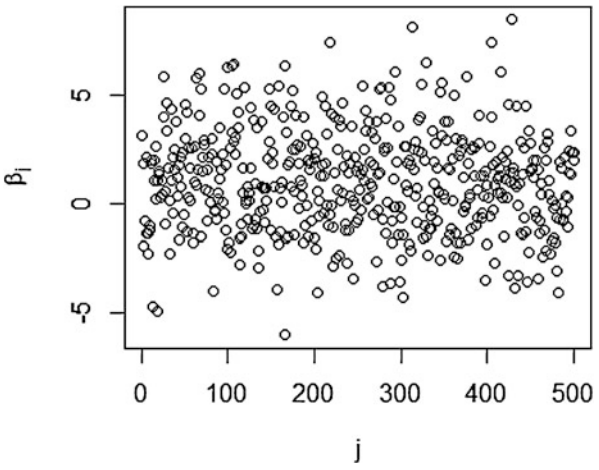

Fig. 3.4 Beta coefficients used in simulation: $\beta_j, j = 1, \ldots, p$

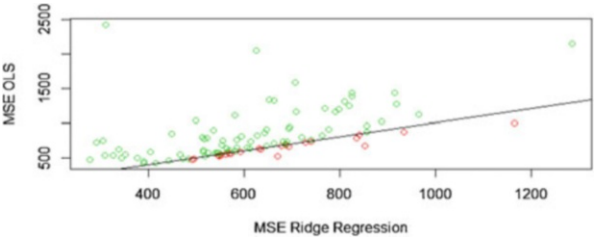

Fig. 3.5 MSE of Ridge regression (MSE RR) versus MSE OLS regression (MSE OLS)

(619.32) of the Ridge approach. In terms of the percentage of simulations in favor of each method, Ridge regression was better in 78 out of 100 simulations, while the OLS was better only in 22 out of 100 simulations. In general, from this small simulation study we obtained more evidence in favor of the Ridge regression method.

The R code used for obtaining this result is the following:

```
############################R code for Example 3######################
rm(list=ls(all=TRUE))
library(mvtnorm)
library(MASS)
source('TR_RR.R')
set.seed(10)
n = 100
p = 500
Var = 0.25*diag(p)+0.75
Tab = data.frame()
betav = rnorm(p,rpois(p,1),2)
plot(betav,xlab=expression(j),ylab=expression(beta[j]))
for(i in 1:100)
{
 X = rmvnorm(n,rep(0,p),Var)
```

```
 dim(X)
 y = 5+X%*%betav + rnorm(n,0,0.5)
 Pos_tr = sample(1:n,n*0.80)
 y_tr = y[Pos_tr]; X_tr = X[Pos_tr,]
 y_tst = y[-Pos_tr]; X_tst = X[-Pos_tr,]
 #Training RR
 TR_RR = Tr_RR_f(y_tr,X_tr,K=5,KG=100,KR=1)
 TR_RR$lamb_o
 lambv = TR_RR$lambv
 plot(log(TR_RR$lambv),TR_RR$iPEv_mean)
 #Prediction RR in testing data
 Pred_RR = Pred_RR_f(y_tst,X_tst,TR_RR)
 Pred_RR$MSEP

 #OLS
 Pred_ols = Pred_ols_f(y_tst,X_tst,y_tr,X_tr)
 Pred_ols

 Tab = rbind(Tab,data.frame(Sim=i,MSEP_RR = Pred_RR$MSEP,
               MSEP_ols = Pred_ols$MSEP))
 cat('i = ', i,'\n')}

Mean_v = colMeans(Tab)
(Mean_v[3]-Mean_v[2])/Mean_v[2]*100

mean(Tab$MSEP_RR<Tab$MSEP_ols)
Pos = which(Tab$MSEP_RR<Tab$MSEP_ols)
mean((Tab$MSEP_ols[Pos]-Tab$MSEP_RR[Pos])/Tab$MSEP_RR[Pos])*100
mean((Tab$MSEP_RR[-Pos]-Tab$MSEP_ols[-Pos])/Tab$MSEP_ols[-Pos])
*100

plot(Tab$MSEP_RR, Tab$MSEP_ols,
   col=ifelse(Tab$MSEP_RR<Tab$MSEP_ols,3,2),
   xlab='MSEP RR', ylab='MSEP OLS')
abline(a=0,b=1)
```

The TR_RR.R script file is the same as the one defined in Example 2 in Appendix 1.

3.6.2 Lasso Regression

Like Ridge regression, the Lasso regression solves the OLS problem but penalizes the residual sum squared in a slightly different way. With the standardized variables, the Lasso estimator of $\boldsymbol{\beta}_s$ is defined as

$$\widetilde{\boldsymbol{\beta}}_s^L(\lambda) = \arg\min_{\mu,\boldsymbol{\beta}_{0s}} \mathrm{PRSS}_\lambda(\boldsymbol{\beta}_s),$$

where now $\text{PRSS}_\lambda(\boldsymbol{\beta}_s) = \sum_{i=1}^{n}\left(y_i - \mu - \sum_{j=1}^{p} x_{ijs}\beta_{js}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_{js}|$ is the $\text{RSS}(\boldsymbol{\beta})$ but penalized by the sum of the absolute regression coefficients. For $\lambda = 0$, the solution is the OLS, while when λ is large, the OLS solutions are shrunken toward 0 (Tibshirani 1996).

Note that for any given values of $\boldsymbol{\beta}_{0s}$, the value of μ that minimizes $\text{PRSS}_\lambda(\boldsymbol{\beta}_s)$ is the sample mean of the responses, $\widetilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i$, the same as the Ridge estimator. However, the rest of the Lasso estimator of $\boldsymbol{\beta}_s$, $\boldsymbol{\beta}_{0s}$, cannot be obtained analytically, so numerical methods are often used.

Although there are efficient algorithms for computing the entire regularization path for the Lasso regression coefficients (Efron et al. 2004; Friedman et al. 2008), here we will describe the coordinate-wise descent given in Friedman et al. (2007). The idea of this method is to successively optimize the $\text{PRSS}_\lambda(\boldsymbol{\beta}_s)$ one parameter at a time (beta coefficient). Holding β_{ks}, $j \neq k$, fixed at their current values $\widetilde{\beta}_{js}(\lambda)$, the value of β_k that minimizes $\text{PRSS}_\lambda(\boldsymbol{\beta}_s)$ is given by

$$\begin{aligned}\widetilde{\beta}^{*}_{ks}(\lambda) &= S\left(\sum_{i=1}^{n} x_{ijs}\left(y_i - \widetilde{y}_i^{(k)}\right), \lambda\right) \\ &= S\left(n\widetilde{\beta}_{ks}(\lambda) + \sum_{i=1}^{n} x_{ijs}(y_i - \widetilde{y}_i), \lambda\right),\end{aligned}$$

where $\widetilde{y}_i^{(k)} = \overline{y} + \sum_{\substack{j=1 \\ j\neq k}}^{p} x_{ijs}\widetilde{\beta}_{js}(\lambda)$ and $S(\beta,\lambda) = \begin{cases} \beta - \lambda & \text{if } \beta > 0 \text{ and } \lambda < |\beta| \\ \beta + \lambda & \text{if } \beta < 0 \text{ and } \lambda < |\beta| \\ 0 & \text{if } \lambda \geq |\beta| \end{cases}$. To obtain the Lasso estimate of $\boldsymbol{\beta}_{0s}$, this process is repeated across all the coefficients until a convergence threshold criterion is reached.

This algorithm can be implemented with the glmnet R package (Friedman et al. 2010) as part of a more general penalty regression (elastic net), which is defined as a combination of the Ridge and Lasso penalties. Due to the structure of the algorithm, this can be used on very large data sets and can benefit from sparsity in the explanatory variables (Friedman et al. 2008).

Equivalently, the Lasso estimator of beta coefficients $\boldsymbol{\beta}_{0s}$ can be defined as

$$\begin{aligned}\widetilde{\boldsymbol{\beta}}^{L}_{0s}(\lambda) &= \underset{\boldsymbol{\beta}_{0s}}{\text{argmin}} \sum_{i=1}^{n}\left(y_i - \overline{y} - \sum_{j=1}^{p} x_{ijs}\beta_{js}\right)^2 \\ \text{subject to } &\sum_{j=1}^{p}|\beta_{js}| \leq t\end{aligned}$$

With this, a graphic representation of the Lasso estimator is like the Ridge (see Fig. 3.6). The nested ellipsoids correspond to contour plots of $\text{RSS}(\boldsymbol{\beta})$ and the green

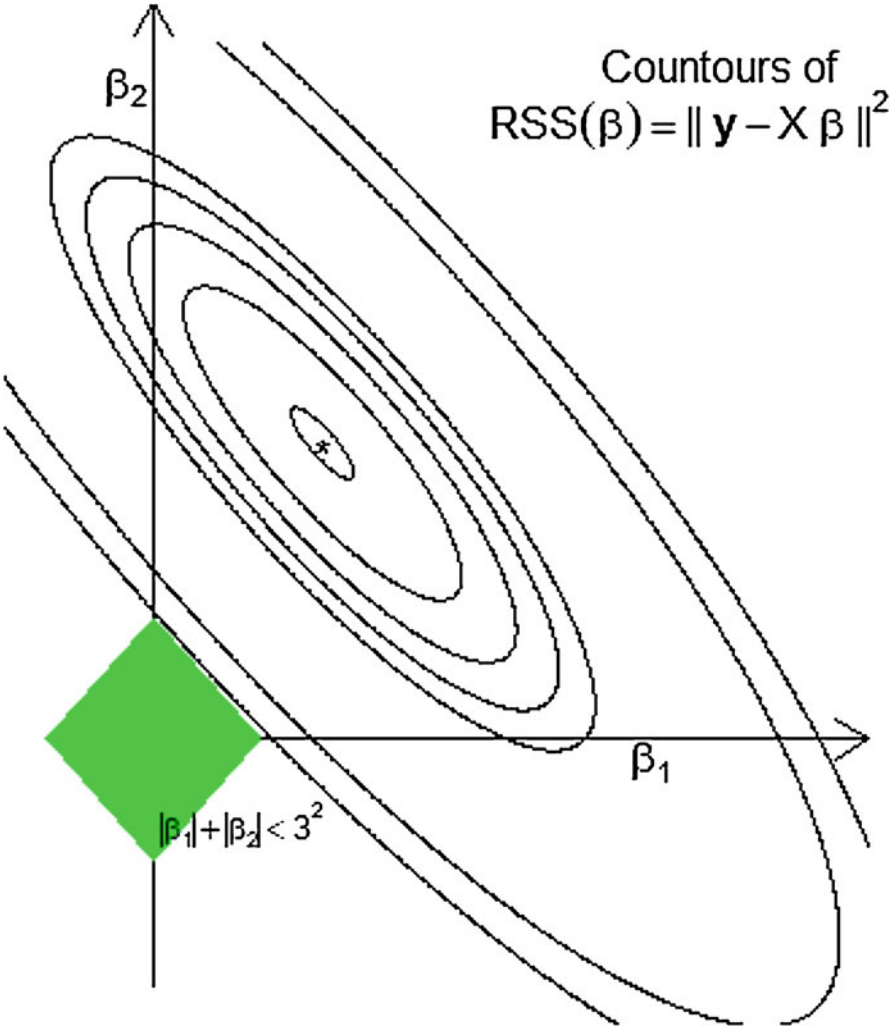

Fig. 3.6 Graphic representation of the Lasso solution of the OLS with restriction $\sum_{j=1}^{p} |\beta_j| < 3^2$. The green region contains the Lasso solution

region is the restriction with $t = 3^2$, which contains the Lasso solution. Indeed, the Lasso solution is the first point of the contours that touches the square, and this will sometimes be in a corner that makes some coefficients zero. Because there are no corners in Ridge regression, this will rarely happen (Tibshirani 1996).

The Lasso estimator can also be derived from a Bayesian perspective. Supposing that the vector of residuals is distributed as $N_n(\mathbf{0}, \sigma^2 \boldsymbol{I}_n)$, like in the Ridge regression case, and assuming that the priors of $\boldsymbol{\beta}_{0s}$ are independent and identically distributed according to Laplace distribution with mean 0 and variance σ_β^2, and adopting a "flat" prior for μ, with known σ^2 and σ_β^2, the posterior distribution of $\boldsymbol{\beta}$ is

$$
\begin{aligned}
& f(\boldsymbol{\beta}_s | \boldsymbol{y}, \boldsymbol{X}_{1s}) \\
& \propto \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}_s\boldsymbol{\beta}_s)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}_s\boldsymbol{\beta}_s)\right] \prod_{i=1}^{n} \exp\left(-\frac{\sqrt{2}}{\sigma_\beta^2}|\beta_{js}|\right) \\
& \propto \exp\left\{-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i - \mu - \sum_{j=1}^{p} x_{ijs}\beta_{js}\right) + \lambda \sum_{j=1}^{p}|\beta_{js}|\right]\right\},
\end{aligned}
$$

where $\lambda = \sqrt{8}\sigma^2/\sigma_\beta^2$. Then, the model of the posterior distribution of $\boldsymbol{\beta}_s$ corresponds to the Lasso estimator described before, $\widetilde{\boldsymbol{\beta}}^L(\lambda)$.

The performance of Lasso regression in terms of prediction error is sometimes comparable to Ridge regression (Hastie et al. 2009). However, as we pointed out before, and based on the nature of the restriction term, for any given value of t, only a subset of the coefficients β_{js} is nonzero, so this gives a sparse solution (Efron et al. 2004).

Example 4 To illustrate Lasso regression, here we considered the data used in Example 2, but instead of using a five-fold cross-validation (5FCV) to explore the behavior of this, we built 100 random splittings of the complete data set: 80% for training and 20% for testing. Figure 3.7 presents a representation of the MSE of the Lasso regression (y-axis) and the MSE corresponding to Ridge regression (x-axis). In 81 out of 100 random splittings, the Ridge regression approach gives a better performance, and in this case, on average, the Lasso regression shows an MSE that is 92.13% greater than the Ridge solution. In the other cases, the Ridge was worse, on average, by 30.91%.

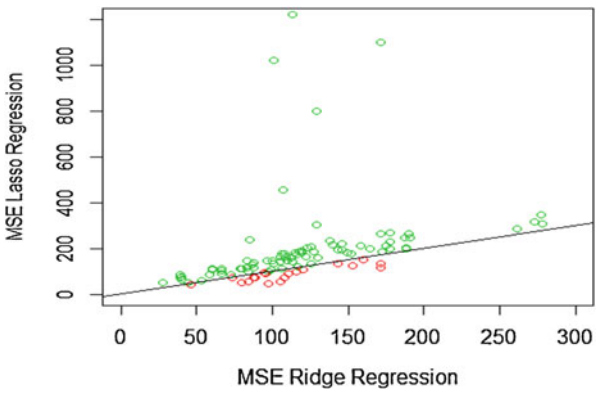

Fig. 3.7 MSE of Ridge regression versus MSE of Lasso regression in 100 random splittings of data: 20% for testing and 80% for training

On average across all the splittings, the performance of the Ridge regression (average = 118.9726 and standard deviation = 50.7193 of MSE) was superior to the Lasso (200.6021 and standard deviation = 222.5494 of MSE) by 68.61%, but this was better than the OLS solution (average = 1609.4635 and standard deviation = 1105.4434 of MSE) by 802.32%, while the Ridge was 1352.80% better than the OLS estimate.

```
#########################R code for Example 4#########################
rm(list=ls())
library(BMTME)
data("WheatMadaToy")
dat_F = phenoMada
dim(dat_F)
dat_F$GID = as.character(dat_F$GID)
G = genoMada
eig_G = eigen(G)
G_0.5 = eig_G$vectors%*%diag(sqrt(eig_G$values))%*%t(eig_G$vectors)
X = G_0.5
y = dat_F$PH
n = length(y)
library(glmnet)
#5FCV
set.seed(3)
K = 5
Tab = data.frame()
set.seed(1)
for(k in 1:100)
{
 Pos_tr = sample(1:n,n*0.8)
 y_tr = y[Pos_tr]; X_tr = X[Pos_tr,]; n_tr = dim(X_tr)[1]
 y_tst = y[-Pos_tr]; X_tst = X[-Pos_tr,]
 #Partition for internal training the model
 Grpv_k = findInterval(cut(sample(1:n_tr,n_tr),breaks=5),1:n_tr)
 #RR
 A_RR = cv.glmnet(X_tr,y_tr,alpha=0,foldid=Grpv_k,type.
measure='mse')
 yp_RR = predict(A_RR,newx=X_tst,s='lambda.min')
 #LR
 A_LR = cv.glmnet(X_tr,y_tr,alpha=1,foldid=Grpv_k,type.
measure='mse')
 yp_LR = predict(A_LR,newx=X_tst,s='lambda.min')
 #OLS
 A_OLS = glmnet(X_tr,y_tr,alpha=1,lambda=0)
 yp_OLS = predict(A_OLS,newx=X_tst)

 Tab = rbind(Tab,data.frame(PT=k,MSEP_RR = mean((y_tst-yp_RR)^2),
             MSEP_LR = mean((y_tst-yp_LR)^2),
             MSEP_OLS = mean((y_tst-yp_OLS)^2)))
 cat('k = ', k,'\n')
}
Tab
```

Now the key components of the just given R code are

1. Hundred random partitions were implemented where each partition is obtained with Pos_tr = sample(1:n,n*0.8), which means that 80% of the data is used for training and 20% for testing, and for each training set, an inner K=5 fold CV is performed to tune the λ hyperparameter.
2. The Grpv_k contains the information of the K=5 fold inner CV implemented to tune the hyperparameter λ.
3. Now we use the cv.glmnet function that is useful for implementing supervised learning methods with cross-validation. This function belongs to the R package glmnet and the input we give to this function is the training set (X_tr, y_tr), alpha=0, that tells glmnet to implement a Ridge regression method, while alpha=1 orders glmnet to implement a Lasso regression. In foldid=Grpv_k we are given training and testing sets to tune the hyperparameter λ, and in type. measure='mse', we are specifying the metric with which we will evaluate the prediction performance of the inner testing sets to be able to choose the best hyperparameter.
4. The function glmnet with lambda=0 implements the OLS estimator.

It is important to point out that Lasso regression performs particularly well when there is a subset of true coefficients that are small or even zero. It doesn't do as well when all of the true coefficients are moderately large; however, in this case, it can still outperform linear regression over a pretty narrow range of (small) λ values.

3.7 Logistic Regression

The logistic regression is a useful and traditional tool used to explain or predict a binary response based on information of explanatory variables. It models the conditional distribution of the response variable as a Bernoulli distribution with the probability of success given by

$$P(Y_i = 1|\boldsymbol{x}_i) = p(\boldsymbol{x}_i;\boldsymbol{\beta}) = \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}.$$

To estimate parameters under logistic regression, suppose that we have a set of data $\left(\boldsymbol{x}_i^{\mathrm{T}}, y_i\right)$, $i = 1, \ldots, n$ (training data), where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$ is a vector of features measurement and y_i is the response measurement corresponding to the ith drawn individual. To obtain the MLE of $\boldsymbol{\beta}$, first we need to build the likelihood function of the parameters of $\boldsymbol{\beta}$. This is given by

$$\begin{aligned} L(\boldsymbol{\beta};\boldsymbol{y}) &= \prod_i^n p(\boldsymbol{x}_i;\boldsymbol{\beta})^{y_i}[1 - p(\boldsymbol{x}_i;\boldsymbol{\beta})]^{1-y_i} = \prod_i^n \left(\frac{p(\boldsymbol{x}_i;\boldsymbol{\beta})}{1 - p(\boldsymbol{x}_i;\boldsymbol{\beta})}\right)^{y_i}[1 - p(\boldsymbol{x}_i;\boldsymbol{\beta})]^1 \\ &= \exp\left(\sum_{i=1}^n y_i\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)\right)\prod_{i=1}^n \frac{1}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}. \end{aligned}$$

and from here the log-likelihood is

$$\ell(\boldsymbol{\beta};\boldsymbol{y}) = \log\left[L(\boldsymbol{\beta};\boldsymbol{y})\right] = \sum_{i=1}^{n} y_i\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right) - \sum_{i=1}^{n} \log\left[1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)\right].$$

Then, because the gradient of the likelihood is given by

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\beta};\boldsymbol{y})}{\partial \boldsymbol{\beta}} &= \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} y_i x_{i1} \\ \vdots \\ \sum_{i=1}^{n} y_i x_{ip} \end{bmatrix} - \begin{bmatrix} \sum_{i=1}^{n} \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)} \\ \sum_{i=1}^{n} \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)} x_{i1} \\ \vdots \\ \sum_{i=1}^{n} \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)} x_{ip} \end{bmatrix} \\ &= \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}) \\ &= \boldsymbol{X}^{\mathrm{T}}[\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta})],\end{aligned}$$

where $\boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}) = [p(\boldsymbol{x}_1;\boldsymbol{\beta}), \ldots, p(\boldsymbol{x}_n;\boldsymbol{\beta})]^{\mathrm{T}}$, the MLE of $\boldsymbol{\beta}$, $\widehat{\boldsymbol{\beta}}$, can be iteratively approximated by using the gradient descent method:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \alpha \boldsymbol{X}^{\mathrm{T}}[\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}_t)].$$

For inferential purposes, we have that the asymptotic distribution of $\widehat{\boldsymbol{\beta}}$ is a multivariate normal distribution with vector mean $\boldsymbol{\beta}$ and variance–covariance matrix, the inverse of the negative of the expected value of the Hessian of the log-likelihood, $E\left\{\left[-\frac{\partial \ell(\boldsymbol{\beta};\boldsymbol{y})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^{\mathrm{T}}}\right]^{-1}\right\} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-1}$ (McCullagh and Nelder 1989). This is because

$$\begin{aligned}\frac{\partial^2 \ell(\boldsymbol{\beta};\boldsymbol{y})}{\partial \beta_j \partial \beta_k} = \frac{\partial \ell(\boldsymbol{\beta};\boldsymbol{y})}{\partial \beta_j} &= -\sum_{i=1}^{n} \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)}{\left[1 + \exp\left(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_0\right)\right]^2} x_{ij} x_{ik} \\ &= -\sum_{i=1}^{n} p(\boldsymbol{x}_i;\boldsymbol{\beta})[1 - p(\boldsymbol{x}_i;\boldsymbol{\beta})] x_{ij} x_{ik}\end{aligned}$$

The Hessian of the log-likelihood is given by

$$\frac{\partial \ell(\boldsymbol{\beta};\boldsymbol{y})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^{\mathrm{T}}} = \boldsymbol{H} = -\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X},$$

where $\boldsymbol{W} = \mathrm{Diag}\{p(\boldsymbol{x}_1;\boldsymbol{\beta})[1 - p(\boldsymbol{x}_1;\boldsymbol{\beta})], \ldots, p(\boldsymbol{x}_n;\boldsymbol{\beta})[1 - p(\boldsymbol{x}_n;\boldsymbol{\beta})]\}$.

Once the parameters have been estimated, the prediction response is obtained from the estimated probabilities: $\widehat{y}_o = 1$ if $p\left(\boldsymbol{x}_o;\widehat{\boldsymbol{\beta}}\right) > 0.5$ and $\widehat{y}_o = 0$ if

$p\left(\boldsymbol{x}_o;\widehat{\boldsymbol{\beta}}\right) \leq 0.5$. Of course, a different threshold to 0.5 can be used and this could be considered as a hyperparameter that needs to be tuned in a similar fashion as the penalty parameter in the Ridge procedure.

It is important to point out that the minimization process of the log-likelihood can be performed using a more efficient iterative technique called the Newton–Raphson technique, which is an iterative optimization technique that uses a local quadratic approximation to the log-likelihood function. The following is the Newton–Raphson iterative equation used to search for the beta coefficients:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \boldsymbol{H}^{-\mathbf{1}}\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}_t)],$$

where $\boldsymbol{H} = -\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}$ is the Hessian matrix whose elements comprise the second derivative of the log-likelihood with regard to the beta coefficients. Therefore, the inverse of the Hessian is $\boldsymbol{H}^{-\mathbf{1}} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X})^{-\mathbf{1}}$. So, the previous equation can be expressed as

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}_t)].$$

It is important to recall that the Hessian is no longer constant since it depends on $\boldsymbol{\beta}$ through the weighting matrix $\boldsymbol{W}$. Also, it is clear that the logistic regression does not have a closed solution due to the nonlinearity of the logistic sigmoid function. It is important to point out that if instead of maximizing the likelihood we minimize the negative of the log-likelihood, the Newton–Raphson equation for updating the beta coefficients is equal to

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{X};\boldsymbol{\beta}_t)].$$

This Newton–Raphson algorithm for logistic regression is known as the iterative reweighted least squares since the diagonal weighting matrix $\boldsymbol{W}$ is interpreted as variances. Another alternative method for estimating the beta coefficients in logistic regression is the Fisher scoring method that is very similar to the Newton–Raphson method just described, but with the difference that instead of using the Hessian ($\boldsymbol{H}$), it uses the expected value of the Hessian matrix, $E(\boldsymbol{H})$.

3.7.1 Logistic Ridge Regression

Like the MLR, when there is strong collinearity, the variance of the MLE is severely affected and the true effects of the explanatory variables could be falsely identified (Lee and Silvapulle 1988). In a similar fashion as for the MLR, this could be judged directly from the asymptotic covariance matrix of $\widehat{\boldsymbol{\beta}}$. Moreover, in a common prediction context, when the number of features is larger than the number of

observations ($p \gg n$), the matrix design is not of full column rank and can cause overfitting, affecting the expected classification error (generalization error) when using the "MLE." One way to avoid overfitting is by replacing the MLE with a regularized MLE as the Ridge MLE estimator of MLR. This is defined as

$$\widetilde{\boldsymbol{\beta}}_s^R(\lambda) = \underset{\boldsymbol{\beta}_s}{\operatorname{argmax}} \left[\ell(\boldsymbol{\beta}_s; \boldsymbol{y}\ ) - \lambda \sum_{j=1}^{p} \beta_{js}^2 \right],$$

where λ is a hyperparameter that has a similar interpretation as in the MLR.

In the literature, there are some algorithms that approximate the Ridge estimation. For example, Genkin et al. (2007) used a cyclic coordinate descent optimization algorithm to approximate this. The one-dimensional optimization problem involved is solved by a modified Newton–Raphson method. Another method was proposed by Friedman et al. (2008) in a more general context. Given the current values of $\widetilde{\boldsymbol{\beta}}_s(\lambda)$, the next update of coordinate β_k is given by

$$\beta_{ks} = \frac{\sum_{i=1}^{n} w_i y_{ij}^* x_{ij}}{\sum_{i=1}^{n} w_i x_{ij}^2 + \lambda}$$

with $y_{ij}^* = y_i^* - \widetilde{\mu}(\lambda) - \sum_{\substack{j=1 \\ j \neq k}}^{p} x_{ijs} \widetilde{\beta}_{js}(\lambda)$ for $k = 1, \ldots, p$, and of μ is given by

$$\mu = \frac{\sum_{i=1}^{n} w_i e_i^*}{\sum_{i=1}^{n} w_i}$$

with $e_i^* = y_i^* - \sum_{j=1}^{p} x_{ijs} \widetilde{\beta}_{js}(\lambda)$, where $y_i^* = \widetilde{\beta}_0(\lambda) + \boldsymbol{x}_i^{\mathrm{T}} \widetilde{\boldsymbol{\beta}}_{0s}(\lambda) + \frac{y_i - p\left(\boldsymbol{x}_i; \widetilde{\boldsymbol{\beta}}_s(\lambda)\right)}{w_i}$ and $w_i = p\left(\boldsymbol{x}_i; \widetilde{\boldsymbol{\beta}}_s(\lambda)\right)\left[1 - p\left(\boldsymbol{x}_i; \widetilde{\boldsymbol{\beta}}_s(\lambda)\right)\right]$, $i = 1, \ldots, n$, are pseudo responses and weights that change across the updates. This can be obtained by maximizing, with respect to β_{ks}, the next quadratic approximation of the penalized likelihood at the current values of $\boldsymbol{\beta}_s$ ($\widetilde{\boldsymbol{\beta}}_s(\lambda)$)

$$\ell(\boldsymbol{\beta}_s; \boldsymbol{y}) - \lambda \sum_{j=1}^{p} \left|\beta_{js}\right| \approx \ell^*(\boldsymbol{\beta}_s; \boldsymbol{y}) - \frac{\lambda}{2} \sum_{j=1}^{p} \beta_{js}^2 + c,$$

where $\ell^*(\boldsymbol{\beta}_s; \boldsymbol{y}) = -\frac{1}{2} \sum_{i=1}^{n} w_i \left(y_i^* - \mu - \sum_{j=1}^{p} x_{ijs} \beta_{js} \right)^2$ is the quadratic approximation of $\ell(\boldsymbol{\beta}_s; \boldsymbol{y})$ at current values of $\boldsymbol{\beta}_s$, $\widetilde{\boldsymbol{\beta}}_s(\lambda)$, and c is a constant that does not depend on $\boldsymbol{\beta}_s$. More details of this implementation (glmnet R package) can be found in Friedman et al. (2008). Note that under this approximation, the beta coefficients can be updated by

$$\widehat{\boldsymbol{\beta}}_s^R(\lambda) = \left(\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}_s + \boldsymbol{\lambda}\boldsymbol{D}\right)^{-1}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{W}\boldsymbol{y},$$

where $\boldsymbol{W} = \mathrm{Diag}(w_1, \ldots, w_n)$ and $\boldsymbol{D}$ is the diagonal matrix as defined before.

3.7.2 *Lasso Logistic Regression*

The Lasso penalization can be applied to other models (Tibshirani 1996). In particular, for logistic regression, the Lasso estimator of $\boldsymbol{\beta}_s$ is defined as

$$\widetilde{\boldsymbol{\beta}}_s^L(\lambda) = \underset{\boldsymbol{\beta}_s}{\mathrm{argmax}}\ell_L(\boldsymbol{\beta}_s;\boldsymbol{y}),$$

where $\ell_L(\boldsymbol{\beta}_s;\boldsymbol{y}) = \ell(\boldsymbol{\beta}_s;\boldsymbol{y}) - \lambda\sum_{j=1}^p |\beta_{js}|$ and is often known as the regularized Lasso likelihood. Numerical methods are also required to obtain this Lasso estimate. There are several possibilities (Genkin et al. 2007), such as non-quadratic programming and iteratively reweighted least squares (Tibshirani 1996), but here we will briefly describe the one proposed by Friedman et al. (2008) and implemented in the glmnent R package. This method consists of applying the coordinate descent procedure to a penalized reweighted least square, which is formed by making a Taylor approximation to the likelihood around the current values of the coefficients. That is, this procedure consists of successively updating the parameters by

$$\widetilde{\boldsymbol{\beta}}_s = \underset{\boldsymbol{\beta}_s}{\mathrm{argmin}}\left(-\ell^*(\boldsymbol{\beta}_s;\boldsymbol{y}) + \frac{\lambda}{2}\sum_{j=1}^p |\beta_{js}|\right),$$

where $\ell^*(\boldsymbol{\beta}_s;\boldsymbol{y}) = -\frac{1}{2}\sum_{i=1}^n w_i\left(y_i^* - \mu - \sum_{j=1}^p x_{ijs}\beta_{js}\right)^2 + c$, $y_i^* = \widetilde{\beta}_0(\lambda) + \boldsymbol{x}_i^{\mathrm{T}}\widetilde{\boldsymbol{\beta}}_{0s}(\lambda) + \frac{y_i - p\left(\boldsymbol{x}_i;\widetilde{\boldsymbol{\beta}}_s(\lambda)\right)}{w_i}$, and $w_i = p\left(\boldsymbol{x}_i;\widetilde{\boldsymbol{\beta}}_s(\lambda)\right)\left[1 - p\left(\boldsymbol{x}_i;\widetilde{\boldsymbol{\beta}}_s(\lambda)\right)\right]$, $i = 1, \ldots, n$, as defined in the Ridge regression case. More details of this implementation can be consulted in Friedman et al. (2008).

Example 5 In this example, we used data corresponding to 40 lines planted with four repetitions. For illustrative purposes, we will use as response a binary variable based on Plant Height. The matrix of features used here was obtained from the genomic relationship ($\boldsymbol{G}$), $\boldsymbol{X} = \boldsymbol{Z}_L\boldsymbol{G}^{1/2}$, where $\boldsymbol{G}^{1/2}$ is the square root matrix of $\boldsymbol{G}$.

The performance of logistic regression, logistic Ridge regression, and Lasso logistic regression for this data set was evaluated across 100 random splittings of the complete data set: 20% for testing (evaluation performance) and 80% for training. The performance was measured by the proportion of cases correctly classified (PCCC) in the testing data. These results are summarized in Table 3.2,

Table 3.2 Performance of the standard, Ridge, and Lasso logistic regression models

Method	PCCC	SD
SLR	0.7284	0.0765
Ridge	0.7240	0.0763
Lasso	0.7206	0.0705

PCCC denotes the proportion of cases correctly classified

where for each method the mean (PCCC) and standard deviation (SD) of the PCCC across the 100 splittings are reported. The table indicates that, on average, the standard logistic (SLR) approach shows slightly better performance than the other two approaches, even better than the Lasso solution. Out of the 100 random partitions, 72, 24, and 4, the SLR, the logistic Ridge regression (LRR), and the logistic lasso regression (LLR) resulted in the higher PCCC value, respectively. However, the difference in the performance of the three methods is not significant because of the large deviation obtained across the different partitions.

The computations were done with the help of the glmnet R package using the following R code:

```
##########################R code for Example 5#######################
load(file ='dat-E3.5.RData')
dat_F = dat$dat_F
dat_F = dat_F[order(dat_F$Rep,dat_F$GID),]
head(dat_F)
G = dat$G
dat_F$y = dat_F$Height
ZL = model.matrix(~0+GID,data=dat_F)
colnames(ZL)
Pos = match(colnames(ZL),paste('GID',colnames(G),sep=''))
max(abs(diff(Pos)))
y = dat_F$y
ei = eigen(G)
X = ZL%*%ei$vectors%*%diag(sqrt(ei$values))%*%t(ei$vectors)
n = length(y)
library(glmnet)
#5FCV
set.seed(1)
Tab = data.frame()
set.seed(1)
for(k in 1:100)
{
 Pos_tr = sample(1:n,n*0.8)
 y_tr = y[Pos_tr] ; X_tr = X[Pos_tr,]; n_tr = dim(X_tr)[1]
 y_tst = y[-Pos_tr]; X_tst = X[-Pos_tr,]
 #Partition for internal training the model
 Grpv_k = findInterval(cut(sample(1:n_tr,n_tr),breaks=5),1:n_tr)
 #RR
 A_RR = cv.glmnet(X_tr,y_tr,family='binomial',
          alpha=0,foldid=Grpv_k,type.measure='class')
 yp_RR = as.numeric(predict(A_RR,newx=X_tst,s='lambda.min',
type='class'))
```

```
 #LR
 A_LR = cv.glmnet(X_tr,y_tr,family='binomial',
         alpha=1,foldid=Grpv_k,type.measure='class')
 yp_LR = as.numeric(predict(A_LR,newx=X_tst,s='lambda.min',
type='class'))
 #SLR
 A_SLR = glmnet(X_tr,y_tr,family='binomial',alpha=0,lambda=0)
 yp_SLR = as.numeric(predict(A_SLR,newx=X_tst,type='class'))

 Tab = rbind(Tab,data.frame(PT=k,PCCC_RR = 1-mean(y_tst!=yp_RR),
               PCCC_LR = 1-mean(y_tst!=yp_LR),
               PCCC_SLR = 1-mean(y_tst!=yp_SLR)))
 cat('k = ', k,'\n')
}
Tab
```

Also, in this R code there are four relevant points:

1. Hundred random partitions were implemented, Pos_tr = sample(1:n,n*0.8), with 80% for training and 20% for testing, and for each training set, an inner K=5 fold CV is performed to tune the λ hyperparameter.
2. Also, here Grpv_k contains the information of the K=5 inner folds to tune the hyperparameter λ.
3. The cv.glmnet function with the following input is used: (a) training set (X_tr, y_tr); (b) with alpha=0 to implement a Ridge regression and alpha=1 to implement a Lasso regression; (c) with foldid=Grpv_k containing training and testing sets to tune the hyperparameter λ; (d) with family='binomial' to implement a logistic regression; and (e) with type.measure='class' as a metric for categorical data to measure the prediction performance of the inner testing set to choose the best value of the hyperparameter λ.
4. The function glmnet with family='binomial' and lambda=0 implements the logistic regression but without penalization.

Appendix 1: R Code for Ridge Regression Used in Example 2

The TR_RR.R script file must contain the following:

```
library(epiR)
source("RR.R")
Tr_RR_f<-function(y_tr,X_tr,K=5,KG=100,KR=1)
{
 n_tr = dim(X_tr)[1]
 X_tr_s = scale2_f(X_tr)
 mu   = mean(y_tr)
 y_tr = y_tr-mu
 #Inner CV
 lambv = lamb_f(X_tr,K=KG,li=1e-7,ls=1-1e-7)
```

```
 iPEv_mean = 0
 for(ir in 1:KR)
 {
  iPE_mat = matrix(0,nr=length(lambv),nc=K)
  MSEP_mat=iPE_mat
  Grpv = findInterval(cut(sample(1:n_tr,n_tr),breaks=K),1:n_tr)
  for(i in 1:K)
  {
   Pos_itr = which(Grpv!=i)
   X_itr  = X_tr_s[Pos_itr,]
   y_itr  = y_tr[Pos_itr];
   y_itst  = y_tr[-Pos_itr];
   dat_itr = data.frame(y=y_itr,X=X_itr)
   n_itr =  dim(X_itr)[1]
   betav_itr = A_RR_f_a_V(y_itr,X_itr,lambv)
   yp_mat   = 0 + (X_tr_s[-Pos_itr,])%*%betav_itr
   iPEv    = colMeans((matrix(y_itst,nr=length(y_itst),
                 nc=length(lambv))-yp_mat)**2)
   iPE_mat[,i] =iPEv
  }
  iPEv_mean = (ir-1)/ir*iPEv_mean + rowMeans(iPE_mat)/ir

 }

 #
 plot(log(lambv),iPEv_mean)
 Pos_o = which.min(iPEv_mean)
 lamb_o = lambv[Pos_o]
 A_RR = RR_f(y=y_tr+mu,X=X_tr,lamb=lamb_o,Intercept = TRUE,Std=TRUE)
 betav_s = A_RR$betav_s
 betav_o = A_RR$betav_o
 list(lamb_o = lamb_o, betav_s =  betav_s,
    betav_o = betav_o,
    lambv=lambv,iPEv_mean=iPEv_mean,X_tr=X_tr,y_tr =y_tr,Grpv=Grpv)
}

Pred_RR_f<-function(y_tst,X_tst,TR_RR)
{
 #betava_RR = TR_RR$betava_RR
 y_tr = TR_RR$y_tr; X_tr = TR_RR$X_tr
 X   = rbind(X_tr,X_tst)
 betav_s = TR_RR$betav_s
 betav_o = TR_RR$betav_o
 yp_tst  = c(cbind(1,X_tst)%*%betav_o)
 plot(y_tst,yp_tst); abline(a=0,b=1)
 MSEP_RR =  mean((y_tst-yp_tst)**2)
 list(MSEP=MSEP_RR, betav_s = betav_s,betav_o = betav_o)
}
Pred_ols_f<-function(y_tst,X_tst,y_tr,X_tr)
{
 #OLS
 p =  dim(X_tr)[2]
 A = lm_f(y_tr,X_tr)
```

```
 sdv  = apply(X_tr,2,sd2_f)
 A_inv = diag(c(1,1/sdv))
 A_inv[1,1] = 1
 A_inv[1,2:(p+1)] = -apply(X_tr,2,mean)/sdv
 betav_s = A$betav_s
 betav_o = A_inv%*%betav_s
 yp_tst_ols = c(cbind(1,X_tst)%*%betav_o)
 MSEP_ols = mean((y_tst-yp_tst_ols)**2)
 list(MSEP=MSEP_ols,betav_s = betav_s)
}
```

The script file accessed by source("RR.R") must contain the following R code:

```
sd2_f<-function(x)
{
 (mean((x-mean(x))**2))^0.5
}
scale2_f<-function(X)
{
 scale(X,center=TRUE,scale = apply(X,2,sd2_f))
}

#RR internal standarized: zij = (xij-\bar{x}_j)/ss_j
#ss_j = mean (xij-\bar{x}_j)^2
RR_f<-function(y,X,lamb = 0, Intercept=TRUE,Std=TRUE)
{
 p   = dim(X)[2]; n = dim(X)[1]
 if(Std == TRUE)
 {
  sdv  = apply(X,2,sd2_f)
  A_inv = diag(c(1,1/sdv))
  A_inv[1,1] = 1
  A_inv[1,2:(p+1)] = -apply(X,2,mean)/sdv
  mu = mean(y)
  X_s = scale2_f(X)
  Xa = X_s
  svd_X = svd(Xa); d1  = svd_X$d
  Gama1 = svd_X$v
  U = svd_X$u
  pa   = dim(Gama1)[2]
  betav_s = c(mu,Gama1%*%(((d1/(d1^2+lamb)))*(t(U)%*%(y-mu))))
  betav_o = A_inv%*%betav_s# betav in orginal scale
  list(betav_s = betav_s,betav_o=betav_o)
 }
 else
 {
  Xa = X
  sdv  = apply(X,2,sd2_f)
  p = dim(X)[2]
  svd_X = svd(Xa)
  d1  = svd_X$d
  Gama1 = svd_X$v; U = svd_X$u
  betav = Gama1%*%((d1/(d1^2+lamb))*(t(U)%*%(y)))#-mean(y)
```

```
  betav
  #tX = t(X)
  #betav = ginv(tX%*%X+lamb*diag(p))%*%tX%*%y
  #betav
 }
}

RR_f_V = Vectorize(RR_f,'lamb')

A_RR_f_a <-function(y_itr,X_itr,lamb)
{
 A = RR_f(y=y_itr,X=X_itr,lamb=lamb,Std=FALSE,Intercept = FALSE)
 A
}
A_RR_f_a_V<-Vectorize(A_RR_f_a,'lamb')

lamb_f<-function(X,K=100,li=0.001,ls=0.999)
{
 Xac = scale2_f(X)
 n = dim(Xac)[1]
 R2v = seq(li,ls,length=K)
 lambv = (1-R2v)/R2v*sum(diag(Xac%*%t(Xac)))/n
 lambv = exp(seq(min(log(lambv)),max(log(lambv)),length=K))
 sort(lambv,decreasing = TRUE)
}

library(MASS)
lm_f<-function(y,X)
{
 p = dim(X)[2]
 sdv  = apply(X,2,sd2_f)
 A_inv = diag(c(1,1/sdv))
 A_inv[1,1] = 1
 A_inv[1,2:(p+1)] = -apply(X,2,mean)/sdv
 X = scale2_f(X); mu = mean(y);
 svd_X = svd(X); d = svd_X$d
 d1  = svd_X$d; Gama1 = svd_X$v; U = svd_X$u
 betav_s = c(mu,Gama1%*%((1/d1)*(t(U)%*%(y-mu))))
 betav_o = A_inv%*%betav_s# betav in orginal scale
 list(betav_s = betav_s,betav_o=betav_o)
}
```

References

Allaire JJ, Chollet F (2019) keras: R Interface to 'Keras'. R package version, 2(4)
Beysolow T II (2017) Introduction to deep learning using R: a step-by-step guide to learning and implementing deep learning models using R. Apress, San Francisco, CA
Burden RL, Faires JD (2011) Numerical analysis. Cengage Learning, Boston, MA
Casella G, Berger RL (2002) Statistical inference. Duxbury, Thomson Learning, Pacific Grove, CA

Christensen P (2011) Plane answers to complex questions: the theory of linear models. Springer Science+Business Media, New York
Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499
Friedman J, Hastie T, Hoeing H, Tibshirani R (2007) Pathwise coordinate optimization. Ann Appl Stat 2(1):302–332
Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9(3):432–441
Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22
Genkin A, Lewis D, Madigan D (2007) Large-scale Bayesian logistic regression for text categorization. Technometrics 49(3):291–304
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. The MIT Press, Cambridge, MA
Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer, New York
Hastie T, Tibshirani R, Wainwright M (2015) Statistical learning with sparsity: the lasso and generalizations. Chapman and Hall/CRC, New York
Haykin S (2009) Neural networks and learning machines. Pearson Prentice Hall, New Jersey, p 909
James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning: with applications in R. Springer Science+Business Media, New York
Lee AH, Silvapulle MJ (1988) Ridge estimation in logistic regression. Commun Stat Simul Comput 178(4):1231–1257. https://doi.org/10.1080/03610918808812723
McCullagh P, Nelder JA (1989) Generalized linear models. Chapman and Hall, London, England
Montgomery DC, Peck EA, Vining GG (2012) Introduction to linear regression. Wiley, Hoboken, NJ
Nocedal J, Wright SJ (2006) Numerical optimization. Springer, New York
Rencher AC, Schaalje GB (2008) Linear models in statistics. Wiley, Hoboken, NJ
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc B 58(1):267–288
Wakefield J (2013) Bayesian and frequentist regression methods. Springer Science+Business Media, New York
Warner B, Misra M (1996) Understanding neural networks as statistical tools. Am Stat 50 (4):284–293

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.