Linear Methods for Regression and Classification

Lecture Notes

Prof. Dr. Peter Filzmoser

Peter.Filzmoser@tuwien.ac.at

Institute of Statistics and Mathematical Methods in Economics

Vienna University of Technology, Austria

Vienna, August 2019

Contents

Chapter 1

Preliminaries: The linear model

The aim of the regression model is to describe the output variable y through a linear combination of one or more input variables $x_1, x_2, \ldots, x_p$. "Linear combination" means that the input variables get first weighed with constants and are then summarized. The result should explain the output variable as good as possible. The multiple linear model has the form

$$y = f(x_1, x_2, \ldots, x_p) + \varepsilon$$

with the linear function

$$f(x_1, x_2, \ldots, x_p) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j. \tag{1.1}$$

The goal is to find a functional relation f which justifies

$$y \approx f(x_1, x_2, \ldots, x_p)$$

and respectively

$$y = f(x_1, x_2, \ldots, x_p) + \varepsilon$$

with the error term ε, which should be as small as possible. More statistical assumptions about the error term will be stated later on. The β_j are unknown parameters or coefficients, which will be estimated from given data. The variables x_j can come from different sources:

- quantitative inputs, for example the height of different people
- transformations of quantitative inputs, such as log, square-root, or square
- basis-expansions, such as $x_2 = x_1^2,\ x_3 = x_1^3$, leading to a polynomial representation
- numeric or "dummy" coding of the levels of qualitative inputs
- interactions between variables, for example $x_3 = x_1 \cdot x_2$

No matter the source of the x_j, the model is linear in the parameters.

Typically we estimate the parameters β_j from a set of training data of the following form

$$\begin{pmatrix} y_1 & x_{11} & x_{12} & \cdots & x_{1p} \\ y_2 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & & & \\ y_i & x_{i1} & x_{i2} & \cdots & x_{ip} \\ \vdots & & & & \\ y_n & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} y_1,\ \boldsymbol{x}_1 \\ y_2,\ \boldsymbol{x}_2 \\ \vdots \\ y_i,\ \boldsymbol{x}_i \\ \vdots \\ y_n,\ \boldsymbol{x}_n \end{pmatrix} \tag{1.2}$$

Each $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ is a feature measurement for the i-th case and y_i is the value of the i-th observation. The estimated parameters are denoted by $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ and by inserting those values in the linear model (1.1) for each observation one gets:

$$\hat{y}_i = f(x_{i1}, x_{i2}, \dots, x_{ip}) = \hat{\beta}_0 + \sum_{j=1}^{p} x_{ij}\hat{\beta}_j$$

The predicted values $\hat{y}_i$ should be as close as possible to the real measured values y_i. The definition of “as close as possible” as well as an estimation of how good the model actually is, will be given in the next chapters.

Let us have a look at the model for every observation $i = 1, \dots, n$. We get

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$$

with the i-th error term ε_i. A request often made is the independence of the error terms from each other and normal distribution with expectation 0 and variance σ^2, thus $\varepsilon_i \sim N(0, \sigma^2)$. Thus, the variance σ^2 has to be the same for every observation, and this way the errors are asked to always show the same spread.

Chapter 2

Comparison of models and model selection

Let us consider a statistical model which combines several input variables $x_1, x_2, \ldots, x_p$ linearly in order to explain an output variable y as good as possible. One question often asked is what "as good as possible" really means and if there would be another model giving an explanation of y as good as the one just found. A lot of times the differences of the quadratic error of the observed values y_i and the estimated values $\hat{y}_i$ are used to measure the performance of a model.Those differences $y_i - \hat{y}_i$ are called *residuals*.

There are different strategies to compare several models: One could be interested in reducing the number of input variables used to explain the output variable since this could simplify the model, making it easier to understand. In addition, the measurements of variables is often expensive, a smaller model would therefore be cheaper. Another strategy is to use all of the input variables and derive a small number of new input variables from them (see Chapter 3). Both strategies are aimed at describing the output variable as good as possible, not just for already measured and available values but also for values acquired in the future.

2.1 Test for several coefficients to be zero

Let us assume that we have 2 models of different size

$$\begin{aligned} M_0 &: \; y = \beta_0 x_0 + \beta_1 x_1 + \cdots + \beta_{p_0} x_{p_0} + \varepsilon \\ M_1 &: \; y = \beta_0 x_0 + \beta_1 x_1 + \cdots\cdots + \beta_{p_1} x_{p_1} + \varepsilon \end{aligned}$$

with $p_0 < p_1$. By simultaneously testing several coefficients to be zero we want to find out if the additional variables $x_{p_0+1}, \ldots, x_{p_1}$ in the full model M_1 provide a significant explanation gain to the smaller model M_0. If, for instance, we would like to find out if a categorical variable with k levels can be excluded from the model, one has to test if all dummy-variables used to represent those k levels can be set to zero.
Rephrasing we get: H_0 : "the small model is true", meaning that the model M_0 is acceptable. Basis for this test is the residual sum of squares

$$\mathrm{RSS}_0 = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p_0} \hat{\beta}_j x_{ij} \right)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

for model M_0 and respectively

$$\mathrm{RSS}_1 = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p_1} \hat{\beta}_j x_{ij} \right)^2$$

for model M_1. This leads to the following test statistic

$$F = \frac{\frac{\mathrm{RSS}_0 - \mathrm{RSS}_1}{p_1 - p_0}}{\frac{\mathrm{RSS}_1}{n - p_1 - 1}} \tag{2.1}$$

The F-statistic measures the change in residual sum of squares per additional parameter in the bigger model, and it is normalized by an estimate of σ^2. Under Gaussian assumptions $\varepsilon_i \sim N(0, \sigma^2)$ and independence, and the null hypothesis that the smaller model is correct, the F statistic will be distributed according to

$$F \sim F_{p_1 - p_0, n - p_1 - 1}.$$

For a better understanding of this test statistic we use the fact that the Total Sum of Squares (TSS) can be expressed by the sum of two variations: the sum of squares explained by the regression (Regression Sum of Squares - RegSS) and the Residual Sum of Squares - RSS:

- Total Sum of Squares TSS $= \sum_{i=1}^{n} (y_i - \overline{y})^2$
- Residual Sum of Squares RSS $= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- Regression Sum of Squares RegSS $= \sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2$
- TSS = RegSS + RSS

For the models M_0 and M_1 we have

$$\begin{aligned} \mathrm{TSS} &= \mathrm{RegSS}_0 + \mathrm{RSS}_0 \\ &= \mathrm{RegSS}_1 + \mathrm{RSS}_1 \end{aligned}$$

which leads to

$$\begin{aligned} \mathrm{RegSS}_1 - \mathrm{RegSS}_0 &= \mathrm{RSS}_0 - \mathrm{RSS}_1 \\ &\geq 0 \quad \text{because } p_0 < p_1 \end{aligned}$$

If $\mathrm{RSS}_0 - \mathrm{RSS}_1$ is large, M_1 explains the data significantly better.

Remark: A test for $H_0 : \beta_0 = \beta_1 = \cdots = \beta_p = 0$ might not give the same result as the tests for the single parameters, especially if the x-variables are highly correlated. Here, one can use "Analysis of Variance (ANOVA)" (similarly also for comparing nested models), which is based on the F-statistic. In doing so, the hypothesis $H_0 : \beta_1 = \cdots = \beta_p = 0$ is tested against $H_1 : \beta_j \neq 0$ for any $j = 1, \ldots, p$. Under H_0, the regression would only give noise. The ANOVA table then is:

	DF	MeanSS
RegSS	p	RegSS/p
RSS	$n - p - 1$	RSS/$(n - p - 1)$

with

$$F_0 = \frac{\text{RegSS}/p}{\text{RSS}/(n-p-1)} = \frac{n-p-1}{p}\frac{R^2}{1-R^2} \sim F_{p,n-p-1},$$

where the above distributional assumptions have to be met.

Section 4.1.2, page 21

2.2 Explained variance

The *multiple R-Square* (coefficient of determination) describes the amount of variance that is explained by the model

$$\begin{aligned} R^2 &= 1 - \frac{\text{RSS}}{\text{TSS}} = \frac{\text{RegSS}}{\text{TSS}} \\ &= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \overline{y})^2} \\ &= \text{Cor}^2(y, \hat{y}) \in [0,1] \end{aligned}$$

The closer R-Square is to 1, the better the fit of the regression. If the model has no intercept, $\overline{y} = 0$ is chosen.
Since the denominator remains constant, R-Square grows with the size of the model. In general we are not interested in the model with the maximum R-Square. Rather, we would like to select that model which leads only to a marginal increase in R-Square with the addition of new variables.

The *adjusted R-Square*, a reduced R-Square value, prevents the effect of getting a bigger R-Square even though the fit gets worse, by including the degrees of freedom in the calculation

$$\tilde{R}^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}$$

Section 4.1.2, page 21

2.3 Information criteria

We now want to approach the analysis of nested models $M_1, \dots, M_q$. The models are ranked with $M_1 \prec M_2 \prec \cdots \prec M_q$, therefore if $M_i \prec M_j$, the model M_i can be seen as a special case of M_j which can be obtained by setting some parameters equal to zero, i.e.

$$\begin{aligned} M_1 &: \; y = \beta_0 + \beta_1 x_1 + \varepsilon \\ M_2 &: \; y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon \end{aligned}$$

Of course we have $M_1 < M_2$, since M_1 is a special case of M_2 where $\beta_2 = 0$.

A possibility to evaluate nested models are information criteria. The most popular criteria are AIC and BIC, which try in different ways to combine the model complexity with a penalty term for the number of parameters used in the model.

2.3.1 Akaike's information criterion (AIC)

The criterion of Akaike is based on the Kullback-Leibler (K-L) information $I(f,g)$. K-L describes the loss of information when occurs when approximating a precise probability distribution $f(x)$ by a probability distributiuon $g(x\,|\,\theta)$. The K-L information is defined as:

$$I(f,g) = \int f(x) \log\left(\frac{f(x)}{g(x\,|\,\theta)}\right) dx \tag{2.2}$$

Alternatively, the K-L information can be interpreted as a distance between the "truth" and a model. For $I(f,g)$ we have:

- $I(f,g) > 0$ if $f(x) \neq g(x\,|\,\theta)$
- $I(f,g) = 0$ if $f(x) = g(x\,|\,\theta)$ almost everywhere

The best model is thus loosing the least information compared to all other models. This is equivalent to minimizing $I(f,g)$ over all g. Since neither f nor θ are known, the K-L information needs to be estimated. This involves the maximized log-likelihood function, which finally leads to the *"Akaike Information Criterion" (AIC)*

$$AIC = -2 \max \log(L(\hat{\theta}|Data)) + 2p,$$

where p is the number of parameters in the model.

In regression, for example, the maximized log-likelihood relates to the residual sum-of-squares (RSS). If σ^2 denotes the (known) residual variance, and n the number of observations, then

$$\text{AIC} = \frac{\text{RSS}}{\sigma^2} + 2p$$

From all estimated nested models one selects now that one with minimal AIC.

Section 4.2.4, page 25

2.3.2 Bayes information criterion (BIC)

Similar to AIC, BIC can be used if the model selection is based on the maximization of the log-likelihood function. The BIC is defined as

$$\begin{aligned} \text{BIC} &= -2 \max \log L + \log(n)p \\ &= \frac{\text{RSS}}{\sigma^2} + \log(n)p \quad \text{if } \sigma^2 \text{ is known} \end{aligned}$$

2.3.3 Mallow's Cp

For known σ^2 the Mallow's C_p is defined as

$$C_p = \frac{\text{RSS}}{\sigma^2} + 2p - n$$

If the residual variance σ^2 is not known, it can be estimated by regression with the full model (using all variables). If a full model cannot be computed (too many variables, collinearity, etc.), a regression can be performed on the relevant principal components (see Section 3.3.1), and the variance is estimated from the resulting residuals. For the "true" model, Cp is approximately p, the number of parameters used in this model, and otherwise greater than p. Thus a model where Cp is approximately p should be selected, and preferably that model with smallest p.

2.4 Resampling methods

After choosing a model which provides a good fit to the training data, we want to model the test data as well, with the requirement $y_{\text{Test}} \approx \hat{f}(x_{\text{Test}})$ ($\hat{f}$ was calculated based on the training data only). One could define a loss function L which measures the error between y and $\hat{f}(x)$, i.e.

$$L\left(y, \hat{f}(x)\right) = \begin{cases} (y - \hat{f}(x))^2 & \text{... quadratic error} \\ |y - \hat{f}(x)| & \text{... absolute error} \end{cases}$$

The test error is the expected predicted value of an *independent* test set

$$\text{Err} = \mathbb{E}\left[L\left(y, \hat{f}(x)\right)\right]$$

With a concrete sample, Err can be estimated using

$$\widehat{\text{Err}} = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{f}(\boldsymbol{x}_i)\right)$$

In case of using the loss function with quadratic error, the estimation of Err is well-known under the name *Mean Squared Error (MSE)*. So, the MSE is defined by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(\boldsymbol{x}_i)\right)^2$$

Usually there is only *one* data set available. The evaluation of the model is then done by resampling methods (i.e. cross validation, bootstrap).

2.4.1 Cross validation

A given sample is randomly divided into q parts. One part is chosen to be the test set, the other parts are defined as training data. The idea is as follows: for the kth part, $k = 1, 2, \ldots, q$, we fit the model to the other $q - 1$ parts of the data and calculate the prediction error of the fitted model when predicting the kth part of the data. In order to avoid complicated notation, $\hat{f}$ denotes the fitted function, computed with the kth part of the data removed. The functions all differ depending on the part left out. The evaluation of the model (calculation of the expected prediction error) is done on the kth part of the data set, i.e. for $k = 3$

1	2	3	4	5	$\cdots$	q
Training	Training	*Test*	Training	Training	$\cdots$	Training

$\hat{y}_i = \hat{f}(\boldsymbol{x}_i)$ represents the prediction for $\boldsymbol{x}_i$, calculated by leaving out the kth part of the data set, with $\boldsymbol{x}_i$ allocated in part k. Since each observation exists only once in each test set, we obtain n predicted values $\hat{y}_i,\ i = 1, \ldots, n$.

The estimated cross validation error is then

$$\widehat{\text{Err}}_{\text{CV}} = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{y}_i\right)$$

Choice of q

The case $q = n$ is known as "leave 1 out cross validation". In this case the fit is computed using all the data except the ith. Disadvantages of this method are

- high computational effort
- high variance due to similarity of the n "training sets".

A 5 fold or 10 fold cross validation, thus $k = 5$ or $k = 10$, should therefore be preferred. With large data sets $n_{\text{Train}}/n_{\text{Test}} = 2/1$ or $1/1$ is often used, this means $\frac{n-n/q}{n/q} = \frac{2}{1}$.

2.4.2 Bootstrap

The basic idea is to randomly draw data sets with replacement from the training data, each sample the same size as the original training set. This is done q times, producing q data sets. Then we refit the model to each of the bootstrap data sets, and examine the behavior of the fits over the q replications. The mean prediction error then is

$$\widehat{\text{Err}}_{\text{Boot}} = \frac{1}{q}\frac{1}{n}\sum_{k=1}^{q}\sum_{i=1}^{n} L\left(y_i, \hat{f}_k(\boldsymbol{x}_i)\right),$$

with $\hat{f}_k$ indicating the function assessed on sample k and $\hat{f}_k(\boldsymbol{x}_i)$ indicating the prediction of observation $\boldsymbol{x}_i$ of the kth data set.

Due to the large overlap in test and training sets, $\widehat{\text{Err}}_{\text{Boot}}$ is frequently too optimistic.

The probability of an observation being included in a bootstrap data set is $1 - (1 - \frac{1}{n})^n \approx 1 - e^{-1} = 0.632$. A possible improvement would be to calculate $\widehat{\text{Err}}_{\text{Boot}}$ only for those observations not included in the bootstrap data set, which is true for about $\frac{1}{3}$ of the observations.

Chapter 3

Linear methods for regression

A *linear* regression model assumes that the regression function $\mathbb{E}(y|x_1, x_2, \ldots, x_p)$ is *linear* in the inputs $x_1, x_2, \ldots, x_p$. Linear models were largely developed in the pre-computer age of statistics, but even in today's computer era there are still good reasons to study and use them since they are the foundation of more advanced methods. Some important characteristics of linear models are:

- they are simple and
- often provide an adequate and
- interpretable description of how the inputs affect the outputs.
- For prediction purposes they can often outperform fancier nonlinear models, especially in situations with
  - small numbers of training data or
  - a low signal-to-noise ratio.
- Finally, linear models can be applied to transformations of the inputs and therefore be used to model nonlinear relations.

3.1 Least Squares (LS) regression

3.1.1 Parameter estimation

The multiple linear regression model has the form

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

with the n observed values

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

the design matrix

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & & & \\ 1 & x_{i1} & x_{i2} & \cdots & x_{ip} \\ \vdots & & & & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} 1,\ \boldsymbol{x}_1 \\ 1,\ \boldsymbol{x}_2 \\ \vdots \\ 1,\ \boldsymbol{x}_i \\ \vdots \\ 1,\ \boldsymbol{x}_n \end{pmatrix}$$

and the error term

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

The parameters we are looking for are the regression coefficients

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}.$$

The most popular estimation method is least squares, in which we choose the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^\top$ which minimize the residual sum of squares (RSS)

$$\begin{aligned} \mathrm{RSS}(\boldsymbol{\beta}) &= \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 \\ &= \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \end{aligned}$$

This approach makes no assumptions about the validity of the model, it simply finds the best linear fit to the data.

<u>How do we minimize the RSS?</u>
Let $\boldsymbol{X}$ denote a $(n \times (p+1))$-matrix which each row being an input vector (with a 1 in the first position). Similarly, let $\boldsymbol{y}$ be the n-vector of outputs in the training data set. Then we can write the residual sum of squares as

$$\begin{aligned} \mathrm{RSS}(\boldsymbol{\beta}) &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \\ &= \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \end{aligned}$$

with the Euclidean norm $\|\cdot\|$. This is a quadratic function in the $p+1$ parameters. Since we are looking for the smallest possible value of RSS, we have a classical minimization problem. Differentiating with respect to $\boldsymbol{\beta}$ yields

$$\begin{aligned} \frac{\partial \mathrm{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \begin{pmatrix} \frac{\partial}{\partial \beta_0}\mathrm{RSS}(\boldsymbol{\beta}) \\ \frac{\partial}{\partial \beta_1}\mathrm{RSS}(\boldsymbol{\beta}) \\ \vdots \\ \frac{\partial}{\partial \beta_p}\mathrm{RSS}(\boldsymbol{\beta}) \end{pmatrix} \\ &= -2\boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \\ \frac{\partial^2 \mathrm{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} &= 2\boldsymbol{X}^\top \boldsymbol{X} \end{aligned}$$

Assuming (for the moment) that $\boldsymbol{X}$ is nonsingular (thus there are at least as many observations as parameters and the observations do not lie in a subspace of lower dimension), and hence $\boldsymbol{X}^\top\boldsymbol{X}$ is positive definite (thus invertible) the solution is a minimum. By setting the first derivative to zero we get

$$\boldsymbol{X}^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) = \boldsymbol{0} \tag{3.1}$$

and then obtain the normal equations

$$(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta}) = \boldsymbol{X}^\top\boldsymbol{y},$$

and their unique solution $\hat{\boldsymbol{\beta}}$

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$$

Now the estimated regression parameters $\hat{\boldsymbol{\beta}}$, that provide the best fit in the case of the RSS minimization can be used for the prediction. If one only wants to predict for the available data $\boldsymbol{x}_i$, the estimated value is

$$\hat{y}_i = \hat{f}(\boldsymbol{x}_i) = (1,\boldsymbol{x}_i)\hat{\boldsymbol{\beta}},$$

and it can be compared to y_i. This can be done for each of the n observations which leads to

$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}}\\ &= \underbrace{\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top}_{\text{Hat matrix } H}\boldsymbol{y}\end{aligned}$$

The matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top$ is sometimes called the “hat matrix” because it puts the hat on $\boldsymbol{y}$.

A geometric interpretation of the least squares estimate is the projection of $\boldsymbol{y}$ on the column space of $\boldsymbol{X}$. We minimize $\text{RSS}(\boldsymbol{\beta}) = \|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|^2$ by choosing $\hat{\boldsymbol{\beta}}$ in order for the residual vector $\hat{\boldsymbol{\varepsilon}} = \boldsymbol{y}-\hat{\boldsymbol{y}}$ to be orthogonal to this subspace. This is also expressed by (3.1). The resulting estimate $\hat{\boldsymbol{y}}$ is hence the orthogonal projection of $\boldsymbol{y}$ onto this subspace. The hat matrix computes the orthogonal projection and hence is also known as a projection matrix.

It might happen that the columns of $\boldsymbol{X}$ are not linearly independent (i.e. if two input variables perfectly correlate), so that $\boldsymbol{X}$ is not of full rank. Then $\boldsymbol{X}^\top\boldsymbol{X}$ is singular and the least squares coefficients $\hat{\boldsymbol{\beta}}$ are not uniquely defined. However, the fitted values $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ are still the projection of $\boldsymbol{y}$ on the column space of $\boldsymbol{X}$, there is just more than one solution. This case appears most often when one or more qualitative inputs are coded in a redundant fashion (recoding mostly eliminates the correlation) or in signal and image analysis, where the number of inputs p can exceed the number of training cases n (the features are typically reduced by filtering or regularization).

Section 4.1.1, page 20

3.1.2 Tests and confidence intervals

Theorem 3.1.2.1 (Gauss-Markov Theorem) *Under the model assumption*

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N_n(\boldsymbol{0},\sigma^2\boldsymbol{I})$$

and the condition that the input variables are fixed, the least squares estimate $\hat{\boldsymbol{\beta}}$ is the best, linear, unbiased estimator (BLUE).

Besides the already known property $\hat{\boldsymbol{\beta}} \sim N_{p+1}(\boldsymbol{\beta},\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1})$ we have furthermore:

- $(n-p-1)\hat{\sigma}^2 \sim \sigma^2\chi^2_{n-p-1}$
- $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ are statistically independent.

Based on these distributional properties we can form tests and confidence intervals for the parameters β_j.
To test the hypothesis $H_0 : \beta_j = 0$, $H_1 : \beta_j \neq 0$, we form

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{d_j}},$$

where d_j is the jth diagonal element of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$. Under the null hypothesis, z_j is $z_j \sim t_{n-p-1}$, and hence a large absolute value of z_j will lead to the rejection of the hypothesis.

If σ would be known, z_j would follow a standard normal distribution:

$$\begin{aligned} z_j &= \frac{\hat{\beta}_j}{\sigma\sqrt{d_j}} \\ z_j &\sim N(0,1) \end{aligned}$$

The difference between the tail quantities of the t-distribution and the standard normal becomes negligible as the sample size increases and so typically the normal quantiles are used. It can also be shown that the z_j match the F-statistic described in Chapter 2 for the comparison of models if only one parameter is tested to be zero. For large n the quantiles of $F_{p_1-p_0,n-p_1-1}$ approach those of the $\chi^2_{p_1-p_0}$.

The test for $\beta_j = 0$ can be used to obtain a confidence interval i.e. for a test at level $\alpha = 0.05$ the critical values are $z_{\alpha/2}, z_{1-\alpha/2} = 1.96$ (if σ is known). The $1-\alpha$ confidence interval for β_j is:

$$[\hat{\beta}_j - z_{1-\alpha/2}\underbrace{\sqrt{d_j}\hat{\sigma}}_{\text{SD}(\hat{\beta}_j)}, \hat{\beta}_j + z_{1-\alpha/2}\sqrt{d_j}\hat{\sigma}]$$

An approximative 95%-confidence interval is:

$$\hat{\beta}_j \pm 2 \cdot \text{SD}(\hat{\beta}_j)$$

Even if the Gaussian error assumption does not hold, this interval will be approximately correct, with its coverage approaching $(1-\alpha)$ for $n \to \infty$.

Section 4.1.2, page 21

Section 4.2.1, page 22

3.2 Variable selection

In the case of large amounts of predictors it makes sense to identify the ones who have the largest influence and to set the ones to zero that are not relevant for the prediction. Thus we eliminate variables that will only explain some details, but we keep those which allow for the major explanation of the response variable.

With variable selection only a subset of all input variables is used, the rest is eliminated from the model. For this subset, the least squares estimate is used for parameter estimation. There are many different strategies to choose a subset.

Possibilities for the optimal choice of regressor variables are:

- *Complete testing* of all possible subsets
  - The computational effort increases with increasing p exponentially and gets infeasible with large p (2^p possible models).
  - This approach is especially time consuming if mixed effects are considered as well.
- *Stepwise algorithms* start with a minimal (respectively maximal) model and add (respectively delete) variables with each step; this approach might only lead to a local optimum.
- The global optimum can be found with *efficient* algorithms (like the "leaps and bound" algorithms). The common idea is the exclusion of whole branches in the graph of all models.

3.2.1 Stepwise algorithms

With data sets with many input variables, a search through all possible subsets becomes infeasible. Following algorithms are used instead:

- *Forward stepwise selection:* starts with the intercept and then sequentially adds into the model the predictor that most improves the fit. The improvement of fit is often based on the F-statistic
$$F = \frac{\text{RSS}(\hat{\boldsymbol{\beta}}) - \text{RSS}(\tilde{\boldsymbol{\beta}})}{\text{RSS}(\tilde{\boldsymbol{\beta}})/(n-k-2)}.$$
$\hat{\boldsymbol{\beta}}$ is the parameter vector of the current model with k parameters, $\tilde{\boldsymbol{\beta}}$ the parameter vector of the model with $k+1$ parameters. Typically, predictors are added sequentially until no predictor produces an F ratio greater than the $90th$ or $95th$ percentile of the $F_{1,n-k-2}$ distribution.
- *Backward stepwise selection:* starts with the full model and sequentially deletes the predictors. It typically uses the same F ratio like forward stepwise selection. Backward selection can only be used when $n > p$ in order to have a well defined model.
- There are also hybrid stepwise strategies that consider both forward and backward moves at each stage, and make the "best" move.

Section 4.2.4, page 25

3.2.2 Best subset regression

The *Best subset regression* finds the subset of size k for each $k \in 0, 1, \ldots, p$ that gives the smallest RSS. An efficient algorithm is the already mentioned *Leaps and Bounds algorithm* in Chapter 2 who is feasible for $p \leq 40$.

Leaps and Bounds algorithm: This algorithm creates a tree model and calculates the RSS (or other criteria) for the particular subsets. In Figure 3.1 the AIC for the first subsets is shown. Subsequently, large branches are eliminated by trying to reduce the RSS. The AIC for the model $x2 + x3 = 20$. Through the elimination of $x2$ or $x3$ we get an AIC of at least 18, since the value can reduce by $2p = 2$ at most (this follows from the formula for the $AIC := n\log(RSS/n) + 2p$). By eliminating a regressor in $x1 + x2$ a smaller AIC (>8) can be obtained. Therefore, the branch $x2 + x3$ is left out in the future analysis.

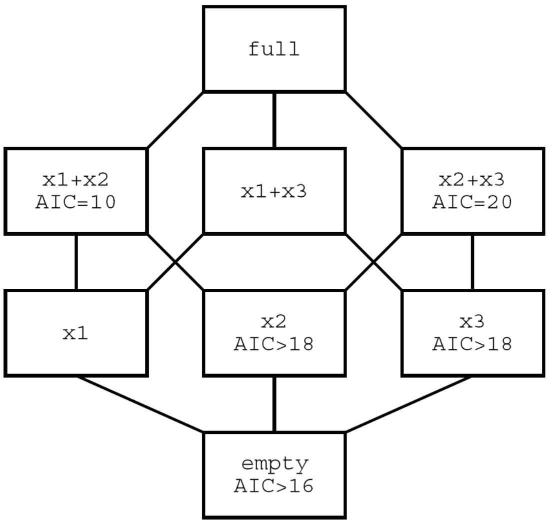

Figure 3.1: Model tree

Section 4.2.5, page 27

3.3 Methods using derived inputs as regressors

In many situations we have a large number of inputs, often highly correlated. For the model

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

with $\boldsymbol{X}$ fixed and $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ we have: $\hat{\boldsymbol{\beta}}_{\text{LS}} \sim N(\boldsymbol{\beta}, \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1})$.

In the case of correlated regressors, $\boldsymbol{X}^\top \boldsymbol{X}$ is almost singular and $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ therefore very large which leads to a numerically unstable $\hat{\boldsymbol{\beta}}$. As a consequence, tests like the one in Section 3.1.2 become unreliable:

$$\begin{aligned} H_0 &: \beta_j = 0 \\ H_1 &: \beta_j \neq 0 \end{aligned}$$

with

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{d_j}},$$

where d_j represents the jth diagonal element of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$.

To avoid this problem, we use methods that use derived input directions. The methods in this section produce a small number of linear combinations $\boldsymbol{z}_k, k = 1, 2, \ldots, q$ of the original inputs $\boldsymbol{x}_j$ which are then used as inputs in the regression. The methods differ in how the linear combinations are constructed.

3.3.1 Principal Component Regression (PCR)

This method looks for transformations of the original data into a new set of uncorrelated variables called principal components. This transformation ranks the new variables according to their importance, which means according to the size of their variance, and eliminates those of least importance. Then a least squares regression on the reduced set of principal components is performed.
Since PCR is not scale invariant, one should scale and center the data. The idea is to construct a new matrix $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{V}$, which consist of linear combinations with the x-variables with coefficients defined by the matrix $\boldsymbol{V}$. The matrix $\boldsymbol{V}$ is a $(p \times p)$-matrix, and thus $\boldsymbol{Z}$ has the same dimension as $\boldsymbol{X}$. The task is to select $\boldsymbol{V}$ such that the columns of $\boldsymbol{Z}$ are uncorrelated and have maximum variance. This can be achieved by a so-called *spectral decomposition* of the covariance matrix $\text{Cov}(\boldsymbol{X})$:

$$\text{Cov}(\boldsymbol{X}) = \boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\top$$

Here, $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p)$ has normed and orthogonal column vectors, i.e. $\boldsymbol{v}_i^\top \boldsymbol{v}_i = 1$, and $\boldsymbol{v}_i^\top \boldsymbol{v}_j = 0$, for $i \neq j = 1, \ldots, p$, and thus $\boldsymbol{V}^\top = \boldsymbol{V}^{-1}$. The matrix $\boldsymbol{A}$ is of diagonal form, $\boldsymbol{A} = Diag(a_1, \ldots, a_p)$.

This solution results from an eigenvalue problem, where $\boldsymbol{v}_i$ are the eigenvectors of $\text{Cov}(\boldsymbol{X})$, and d_i are the corresponding eigenvalues: $a_1 \geq a_2 \ldots \geq a_p$. If $\text{Cov}(\boldsymbol{X})$ is positive definite, all eigenvalues are real, non negative numbers.

The columns of $\boldsymbol{Z}$ are are called *principal components*, and we obtain:

$$\text{Cov}(\boldsymbol{Z}) = \boldsymbol{V}^\top \text{Cov}(\boldsymbol{X})\boldsymbol{V} = \boldsymbol{A}$$

Thus, the variance of the ith principal component is equal to the eigenvalue a_i; the variances are ranked in descending order.

In the following we will use the first q $(1 \leq q < p)$ principal components for regression. The regression model can first be rewritten by using *all* the principal components:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad (3.2) \\ &= \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^\top\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\ &= \boldsymbol{Z}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \end{aligned}$$

The new regression coefficients in this model are denoted by $\boldsymbol{\theta}$, and we have $\boldsymbol{\theta} = \boldsymbol{V}^\top\boldsymbol{\beta}$ and $\boldsymbol{\beta} = \boldsymbol{V}\boldsymbol{\theta}$. If we want to use for regression only $q < p$ principal components, then we obtain by the above relation

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{Z}_{1:q}\boldsymbol{\theta}_{1:q} + \boldsymbol{Z}_{q+1:p}\boldsymbol{\theta}_{q+1:p} + \boldsymbol{\varepsilon} \\ &= \boldsymbol{Z}_{1:q}\boldsymbol{\theta}_{1:q} + \tilde{\boldsymbol{\varepsilon}}. \end{aligned}$$

In this reduced model we obtain with LS estimation

$$\hat{\boldsymbol{\theta}}_{1:q} = (\boldsymbol{Z}_{1:q}^\top \boldsymbol{Z}_{1:q})^{-1}\boldsymbol{Z}_{1:q}^\top \boldsymbol{y}.$$

If the regression coefficients should be interpreted in terms of the original variables, then a back-transformation is possible by using the above relation,

$$\hat{\tilde{\boldsymbol{\beta}}} = \boldsymbol{V}_{1:q}\hat{\boldsymbol{\theta}}_{1:q}.$$

Note that $\tilde{\tilde{\boldsymbol{\beta}}}$ does no longer correspond to the LS estimator from Equation (3.2), except if the first q principal components explain all the information of $\boldsymbol{X}$.

Section 4.3.2, page 31

3.3.2 Partial Least Squares (PLS) regression

This technique also constructs a set of linear combinations of the inputs for regression, but unlike principal components regression it uses $\boldsymbol{y}$ in addition to $\boldsymbol{X}$ for this construction. We assume that $\boldsymbol{X}$ is centered and – depending on the application – also scaled. Instead of estimating the parameters $\boldsymbol{\beta}$ in the linear model

$$y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i$$

we estimate the parameters $\boldsymbol{\gamma}$ in the so-called latent variable model

$$y_i = \boldsymbol{t}_i^\top \boldsymbol{\gamma} + \tilde{\varepsilon}_i.$$

We assume:

- The new coefficients $\boldsymbol{\gamma}$ are of dimension $q \leq p$.
- The values $\boldsymbol{t}_i$ are arranged as rows in an $(n \times q)$ score matrix $\boldsymbol{T}$.
- Due to the reduced dimension, the regression of $\boldsymbol{y}$ on $\boldsymbol{T}$ should be more stable.
- $\boldsymbol{T}$ can not be observed directly; we obtain each column of $\boldsymbol{T}$ sequentially, for $k = 1, 2, \ldots, q$, by using the PLS criterion

  $$\boldsymbol{w}_k = \underset{\boldsymbol{w}}{\operatorname{argmax}} \operatorname{Cov}(\boldsymbol{y}, \boldsymbol{X}\boldsymbol{w})$$

  under the constraints $\|\boldsymbol{w}_k\| = 1$ and $\operatorname{Cov}(\boldsymbol{X}\boldsymbol{w}_k, \boldsymbol{X}\boldsymbol{w}_j) = 0$ for $1 \leq j < k$. The vectors $\boldsymbol{w}_k$ with $k = 1, 2, \ldots, q$ are called *loadings*, and they are collected in the columns of the matrix $\boldsymbol{W}$. The score matrix is then

  $$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W},$$

  and $\boldsymbol{y}$ can be written as:

  $$\begin{aligned} \boldsymbol{y} &= \boldsymbol{T}\boldsymbol{\gamma} + \tilde{\boldsymbol{\varepsilon}} \\ &= (\boldsymbol{X}\boldsymbol{W})\boldsymbol{\gamma} + \tilde{\boldsymbol{\varepsilon}} \\ &= \boldsymbol{X}\underbrace{(\boldsymbol{W}\boldsymbol{\gamma})}_{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\varepsilon}} \approx \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \end{aligned}$$

In other words, PLS does a regression on a weighted version of $\boldsymbol{X}$ which contains incomplete or partial information (thus the name of the method). The additional usage of the least squares method for the fit leads to the name *Partial Least Squares.*
Since PLS uses also $\boldsymbol{y}$ to determine the PLS-directions, this method is supposed to have better prediction performance than for instance PCR. In contrast to PCR, PLS is looking for directions with high variance and large correlation with $\boldsymbol{y}$.

Section 4.3.3, page 32

3.4 Shrinkage methods

Shrinkage methods keep all variables in the model and assign different (continuous) weights. In this way we obtain a smoother procedure with a smaller variability.

3.4.1 Ridge regression

Ridge regression shrinks the coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares,

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{3.3}$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk towards zero (and towards each other).

An equivalent way to write the ridge problem is

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}$$

under the constraint

$$\sum_{j=1}^{p} \beta_j^2 \leq s,$$

which makes explicit the size constraint on the parameters. By penalizing the RSS we try to avoid that highly correlated regressors (e.g. x_j and x_k) cancel each other. An especially large positive coefficient β_j can be canceled by a similarly large negative coefficient β_k. By imposing a size constraint on the coefficients this phenomenon can be prevented.

The ridge solutions $\hat{\boldsymbol{\beta}}_{\text{Ridge}}$ are *not* equivariant for different scaling of the inputs, and so one normally standardizes the inputs. In addition, notice that the intercept β_0 has been left out of the penalty term. Penalization of the intercept would make the procedure depend on the origin chosen for $\boldsymbol{y}$; that is adding a constant c to each of the targets y_i would *not* simply result in a shift of the predictions by the same amount c.

We center the x_{ij}, each x_{ij} gets replaced by $x_{ij} - \overline{x}_j$ and estimate β_0 by $\overline{y} = \sum_{i=1}^{n} y_i/n$. The remaining coefficients get estimated by a ridge regression *without* intercept, hence the matrix $\boldsymbol{X}$ has p rather than $p+1$ columns. Rewriting (3.3) in matrix form,

$$\begin{aligned} \text{RSS}(\lambda) &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^\top\boldsymbol{\beta} \quad \text{the solutions become} \\ \hat{\boldsymbol{\beta}}_{\text{Ridge}} &= (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \end{aligned}$$

$\boldsymbol{I}$ is the $(p \times p)$ identity matrix. Advantages of the just described method are:

- With the choice of quadratic penalty $\boldsymbol{\beta}^\top\boldsymbol{\beta}$, the resulting ridge regression coefficients are again a linear function of $\boldsymbol{y}$.
- The solution adds a positive constant to the diagonal of $\boldsymbol{X}^\top\boldsymbol{X}$ before inversion. This makes the problem nonsingular, even if $\boldsymbol{X}^\top\boldsymbol{X}$ is not of full rank. This was the main motivation of its introduction around 1970.

- The effective degrees of freedom are

$$\mathrm{df}(\lambda) = \mathrm{tr}(\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^\top),$$

  thus

$$\begin{aligned} \text{for } \lambda = 0 \quad &\Rightarrow \quad \mathrm{df}(\lambda) = \mathrm{tr}(\boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}) = \mathrm{tr}(\boldsymbol{I}_p) = \mathrm{p} \\ \lambda \to \infty \quad &\Rightarrow \quad \mathrm{df}(\lambda) \to 0 \end{aligned}$$

In the case of orthogonal inputs, the ridge coefficients are just a scaled version of the least squares estimates, that is $\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \gamma \hat{\boldsymbol{\beta}}$ with $0 \le \gamma \le 1$.

Section 4.4.1, page 33

3.4.2 Lasso Regression

The lasso is a shrinkage method like ridge, but L_1 norm rather than the L_2 norm is used in the constraints. The lasso is defined by

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

with the constraint

$$\sum_{j=1}^{p} |\beta_j| \le s.$$

Just as in ridge regression we standardize the data. The solution for $\hat{\beta}_0$ is $\overline{y}$ and thereafter we fit a model without an intercept.

Lasso and ridge differ in their penalty term. The lasso solutions are nonlinear and a quadratic programming algorithm is used to compute them. Because of the nature of the constraint, making s sufficiently small will cause some of the coefficients to be exactly 0. Thus the lasso does a kind of continuous subset selection. If s is chosen larger than $s_0 = \sum_{j=1}^{p} |\hat{\beta}_j|$ (where $\hat{\beta}_j$ is the least squares estimate), then the lasso estimates are the least squares estimates. On the other hand, for $s = s_0/2$, the least squares coefficients are shrunk by about 50% on average. However, the nature of the shrinkage is not obvious. Like the subset size in subset selection, or the penalty in ridge regression, s should be adaptly chosen to minimize an estimate of expected prediction error.

Figure 3.2 visualizes the difference in the penalties in case of two parameters. The distribution of the residual sum-of-squares is visualized by the elliptical contours, which are centered at the least-squares estimator. The blue regions refer to the contours of the contraints, left for Lasso ($|\beta_1| + |\beta_2| \le s$), and right for Ridge ($\sqrt{\beta_1^2 + \beta_2^2} \le s$). It is clear that Lasso will more likely lead to a solution where regression coefficients are exactly zero.

Section 4.4.2, page 35

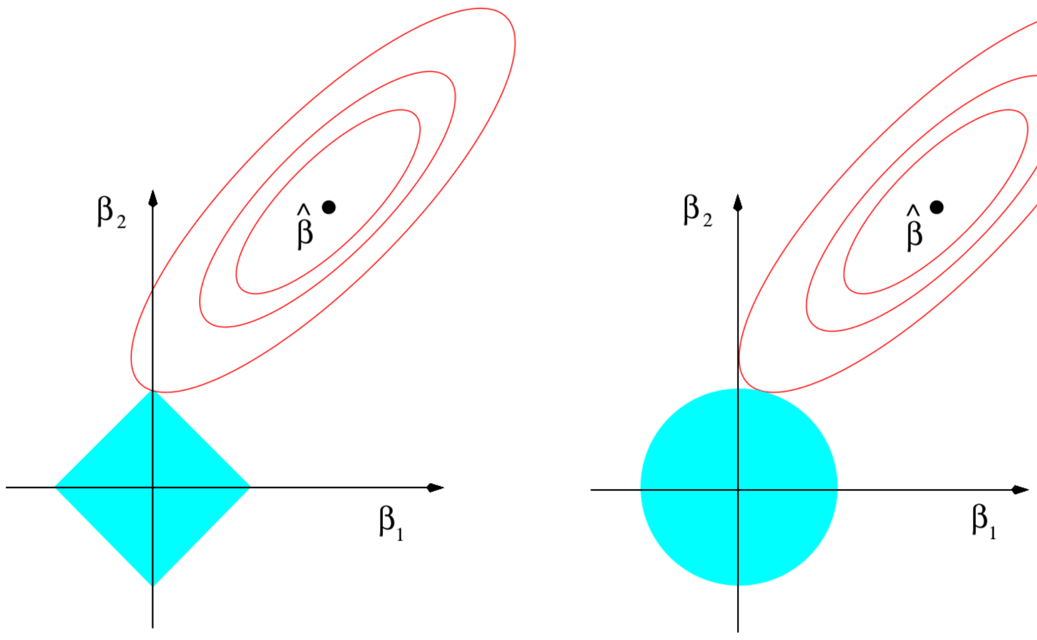

Figure 3.2: Penalties for Lasso (left) and Ridge (right) regression.

Chapter 4

Linear methods in R

4.1 Least Squares (LS) regression in R

4.1.1 Parameter estimation

Here we simulate a data set with a response and 3 explanatory variables. Therefore we know the true regression coefficients, $\beta_0 = 0$, $\beta_1 = 1$, $\beta_2 = 2$ and $\beta_3 = 0$. Accordingly, the variable x_3 has no predictive meaning, and should not be used in the model.

- *Generation of the data*

```
> set.seed(123)
> x <- matrix(runif(60), ncol = 3)
> y <- x %*% c(1, 2, 0) + 0.1 * rnorm(20)
> colnames(x) <- paste("x", 1:3, sep = "")
> d <- data.frame(x, y = y)
> plot(d,pch=4,col=4)
```

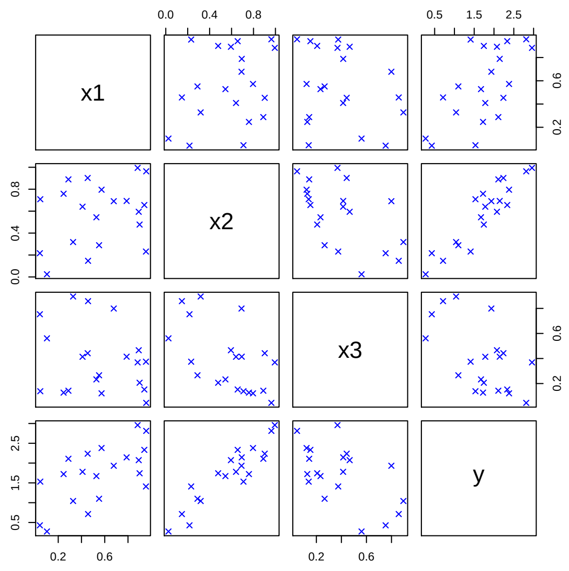

Figure 4.1: Plot of the generated data used for multiple linear regression.

- *Model using only a constant term*

```
> lm0 <- lm(y~1, data = d)
> lm0

Call:
lm(formula = y ~ 1, data = d)

Coefficients:
(Intercept)
       1.72
```

LS regression is computed by **lm()**. The estimated value of the intercept is $\beta_0 = 1.72$.

- *Model with one explanatory variable*

```
> lm1 <- lm(y~x1, data = d)
> lm1

Call:
lm(formula = y ~ x1, data = d)

Coefficients:
(Intercept)           x1
     0.9157       1.4600
```

- *Fit of a full model*

```
> lm3 <- lm(y~x1+x2+x3, data = d)
> lm3

Call:
lm(formula = y ~ x1 + x2 + x3, data = d)

Coefficients:
(Intercept)           x1           x2           x3
    0.09585      0.91834      1.99804     -0.08761
```

The coefficients from the full model are close to the true coefficients. However, we would prefer a model that excludes x_3, so with $\beta_3 = 0$.

4.1.2 Tests and confidence intervals

- *Testing the coefficients for significance*

```
> summary(lm3)

Call:
lm(formula = y ~ x1 + x2 + x3, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.11566 -0.06133 -0.01260  0.06785  0.18004

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.09585    0.08200   1.169    0.260
x1           0.91834    0.06623  13.867 2.47e-10 ***
x2           1.99804    0.08453  23.637 7.18e-14 ***
x3          -0.08761    0.09060  -0.967    0.348
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 0.08621 on 16 degrees of freedom
Multiple R-squared:  0.9882,        Adjusted R-squared:  0.986
F-statistic: 446.5 on 3 and 16 DF,  p-value: 1.251e-15
```

- The t statistic (see Section 3.1.2) of x_1 and x_2 is highly significant and the p-value of each variable is below 0.05. Therefore, both variables have a great impact on the explanation of the regressor and the null hypothesis can be rejected. The regressor x_3 provides no significant additional contribution.
- The model provides a good fit (R squared, see Section 2.2), 98.82% of the variance of y can be explained by the model. The value 98.6% of the adjusted R squared is very high as well.

-
  ```
  > qf(0.95, 3, 16)
  [1] 3.238872
  ```

  The value of the “F statistic” (see Section 2.1) of 446.5 is larger than the F quantile $F_{3,16;0.95} = 3.24$, therefore the null hypothesis $\beta_i = 0, \forall\ i = 1, \ldots, p$ can be rejected. This could also be concluded by the p-value that is close to 0.
- The test statistic from above can be used for the calculation of a confidence interval for $\hat{\beta}_j$ (see Section 3.1.2). From the approximation of the 95% confidence interval, we obtain for $\hat{\beta}_1$ the interval

  $$0.91834 \pm 2 * 0.06623 = [0.78, 1.06]$$

  and for $\hat{\beta}_3$

  $$-0.08761 \pm 2 * 0.09060 = [-0.27, 0.09]$$

  The interval for $\hat{\beta}_1$ does not include zero, and thus the null hypothesis can be rejected at a 95% level. The interval for $\hat{\beta}_3$ includes zero, which confirms the acceptance of the null hypothesis due to a p-value of 0.348.

4.2 Variable selection in R

4.2.1 Model comparison with anova()

```
> anova(lm3)
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq  F value    Pr(>F)
x1         1 3.9799  3.9799 535.4639 9.991e-14 ***
x2         1 5.9693  5.9693 803.1073 4.199e-15 ***
x3         1 0.0070  0.0070   0.9351    0.3479
Residuals 16 0.1189  0.0074
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

An F-test (see Section 2.1) is computed for every additional explanatory variable, starting with the empty model and following the order of the formula. Regressor x_3 does not improve the fit of the model and can be left out.

```
> lm2 <- lm(y~x1+x2, data=d)
> anova(lm0, lm1, lm2, lm3)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x1
Model 3: y ~ x1 + x2
Model 4: y ~ x1 + x2 + x3
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1     19 10.0751
2     18  6.0951  1    3.9799 535.4639 9.991e-14 ***
3     17  0.1259  1    5.9693 803.1073 4.199e-15 ***
4     16  0.1189  1    0.0070   0.9351    0.3479
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here several nested models (with increasing size) are compared in the specified order. This allows simultaneous testing of the significance of more than one parameter. Here, again, model `lm3` does not improve the fit.

4.2.2 Body fat data

- *Scanning of the data and explanation of the variables*

```
      > library("UsingR")
      > data(fat)
      > fat <- fat[-c(31,39,42,86), -c(1,3,4,9)]# strange values, not use all variables
      > attach(fat)
```

The data set “fat” consists of 15 physical measurements of 251 men. The data can be found in the `library(UsingR)`.

  - `body.fat`: percentage of body-fat calculated by Brozek’s equation
  - `age`: age in years
  - `weight`: weight (in pounds)
  - `height`: height (in inches)
  - `BMI`: adiposity index
  - `neck`: neck circumference (cm)
  - `chest`: chest circumference (cm)
  - `abdomen`: abdomen circumference (cm)
  - `hip`: hip circumference (cm)
  - `thigh`: thigh circumference (cm)
  - `knee`: knee circumference (cm)
  - `ankle`: ankle circumference (cm)
  - `bicep`: extended biceps circumference (cm)
  - `forearm`: forearm circumference (cm)
  - `wrist`: wrist circumference (cm)

To measure the percentage of body-fat in the body, an extensive (and expensive) underwater technique has to be performed. The goal here is to establish a model which allows the prediction of the percentage of body-fat with easily measurable and collectible variables in order to avoid the underwater procedure. Nowadays, a new, very effortless method called bio-impedance analysis provides a reliable method to determine the body-fat percentage.

4.2.3 Full model

For model evaluation later on, we first split the data randomly into training (2/3) and test data (1/3). The models will be built with the training data, and the evaluation is based on the test data.

```
> # randomly split into training and test data:
> set.seed(123)
> n <- nrow(fat)
> train <- sample(1:n,round(n*2/3))
> test <- (1:n)[-train]
> model.lm <- lm(body.fat~., data = fat, subset=train)
> summary(model.lm)

Call:
lm(formula = body.fat ~ ., data = fat, subset = train)

Residuals:
    Min      1Q  Median      3Q     Max
-9.6967 -2.5370 -0.3181  2.4634  8.9503

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.32010   42.43522  -0.125   0.9004
age          0.04946    0.03886   1.273   0.2051
weight      -0.03714    0.11980  -0.310   0.7570
height       0.02268    0.58969   0.038   0.9694
BMI          0.46329    0.86154   0.538   0.5916
neck        -0.16060    0.26965  -0.596   0.5523
chest       -0.13454    0.12691  -1.060   0.2908
abdomen      0.83996    0.11375   7.384 9.84e-12 ***
hip         -0.34536    0.16922  -2.041   0.0430 *
thigh        0.19863    0.17014   1.167   0.2449
knee        -0.01277    0.28098  -0.045   0.9638
ankle       -0.34105    0.44172  -0.772   0.4413
bicep        0.11665    0.18273   0.638   0.5242
forearm      0.29547    0.20976   1.409   0.1610
wrist       -1.32879    0.66187  -2.008   0.0465 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.955 on 150 degrees of freedom
Multiple R-squared:  0.7447,        Adjusted R-squared:  0.7208
F-statistic: 31.25 on 14 and 150 DF,  p-value: < 2.2e-16
```

The coefficients for `abdomen`, `hip` and `wrist` have a p-value below 0.05, and therefore the null hypothesis $\beta_i = 0$ should be rejected for the corresponding variables. Due to the very small p-value of the F-statistic, the null hypothesis $\beta_i = 0, \ \forall \ i = 1, \ldots, p$ should be rejected as well. With an R squared = 0.7447 we can assume that the model provides a good fit. Some measures for the prediction performance like R^2 or MSE for the test data can be computed.

```
> pred.lm <- predict(model.lm,newdata = fat[test,])
> cor(fat[test,"body.fat"],pred.lm)^2 # R^2 for test data

[1] 0.7414579

> mean((fat[test,"body.fat"]-pred.lm)^2) # MSE_test

[1] 16.62058
```

Figure 4.2 show the measured versus the predicted response variable for the test data, resulting from the full model.

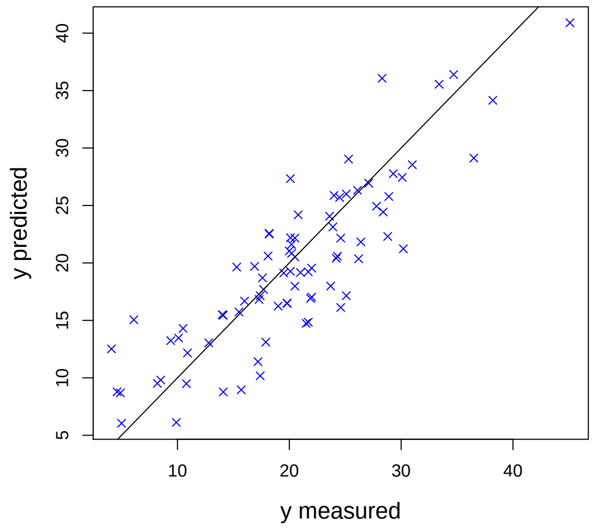

Figure 4.2: Prediction resulting from full model for test data.

4.2.4 Stepwise selection - automatic model search

– *Stepwise selection with* **drop1()**

```
> drop1(model.lm, test="F")
Single term deletions

Model:
body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist
        Df Sum of Sq    RSS    AIC F value    Pr(>F)
<none>               2346.2 468.01
age      1     25.34 2371.5 467.78  1.6200   0.20506
weight   1      1.50 2347.7 466.11  0.0961   0.75698
height   1      0.02 2346.2 466.01  0.0015   0.96938
BMI      1      4.52 2350.7 466.32  0.2892   0.59155
neck     1      5.55 2351.7 466.40  0.3547   0.55234
chest    1     17.58 2363.7 467.24  1.1239   0.29079
abdomen  1    852.89 3199.0 517.17 54.5288 9.839e-12 ***
hip      1     65.15 2411.3 470.53  4.1652   0.04302 *
thigh    1     21.32 2367.5 467.50  1.3629   0.24489
knee     1      0.03 2346.2 466.01  0.0021   0.96382
ankle    1      9.32 2355.5 466.66  0.5961   0.44127
bicep    1      6.37 2352.5 466.45  0.4075   0.52422
forearm  1     31.03 2377.2 468.18  1.9841   0.16103
wrist    1     63.04 2409.2 470.38  4.0306   0.04648 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(update(model.lm,.~.-knee))
Call:
lm(formula = body.fat ~ age + weight + height + BMI + neck +
    chest + abdomen + hip + thigh + ankle + bicep + forearm +
    wrist, data = fat, subset = train)

Residuals:
    Min      1Q  Median      3Q     Max
-9.6903 -2.5408 -0.3137  2.4703  8.9383
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.30457   42.29339  -0.125   0.9004
age          0.04883    0.03620   1.349   0.1794
weight      -0.03730    0.11936  -0.312   0.7551
height       0.01955    0.58373   0.033   0.9733
BMI          0.46113    0.85739   0.538   0.5915
neck        -0.15963    0.26790  -0.596   0.5522
chest       -0.13417    0.12622  -1.063   0.2895
abdomen      0.84015    0.11329   7.416 8.09e-12 ***
hip         -0.34589    0.16825  -2.056   0.0415 *
thigh        0.19643    0.16258   1.208   0.2288
ankle       -0.34570    0.42828  -0.807   0.4208
bicep        0.11683    0.18208   0.642   0.5221
forearm      0.29542    0.20907   1.413   0.1597
wrist       -1.32794    0.65942  -2.014   0.0458 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.942 on 151 degrees of freedom
Multiple R-squared:  0.7447,        Adjusted R-squared:  0.7227
F-statistic: 33.88 on 13 and 151 DF,  p-value: < 2.2e-16
```

Elimination of the least significant variable, in this case `knee` is excluded from the model. The R squared (and adjusted R squared) do not change, the fit remains the same.

– *Automatic model search with* **step()**

```
> model.lmstep <- step(model.lm)
Start:  AIC=468.01
body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist

          Df Sum of Sq    RSS    AIC
- height   1      0.02 2346.2 466.01
- knee     1      0.03 2346.2 466.01
- weight   1      1.50 2347.7 466.11
- BMI      1      4.52 2350.7 466.32
- neck     1      5.55 2351.7 466.40
- bicep    1      6.37 2352.5 466.45
- ankle    1      9.32 2355.5 466.66
- chest    1     17.58 2363.7 467.24
- thigh    1     21.32 2367.5 467.50
- age      1     25.34 2371.5 467.78
<none>                 2346.2 468.01
- forearm  1     31.03 2377.2 468.18
- wrist    1     63.04 2409.2 470.38
- hip      1     65.15 2411.3 470.53
- abdomen  1    852.89 3199.0 517.17

Step:  AIC=466.01
body.fat ~ age + weight + BMI + neck + chest + abdomen + hip +
    thigh + knee + ankle + bicep + forearm + wrist

          Df Sum of Sq    RSS    AIC
- knee     1      0.03 2346.2 464.01
- weight   1      5.08 2351.3 464.37
- neck     1      5.54 2351.7 464.40
- bicep    1      6.38 2352.6 464.46
- ankle    1      9.43 2355.6 464.67
- chest    1     17.60 2363.8 465.24
- thigh    1     21.43 2367.6 465.51
- age      1     25.32 2371.5 465.78
- BMI      1     28.47 2374.6 466.00
<none>                 2346.2 466.01
- forearm  1     31.11 2377.3 466.18
- wrist    1     63.20 2409.4 468.39
- hip      1     65.24 2411.4 468.53
- abdomen  1    855.35 3201.5 515.30

                          ⋮

Step:  AIC=457.89
body.fat ~ age + BMI + chest + abdomen + hip + forearm + wrist

          Df Sum of Sq    RSS    AIC
<none>                 2402.0 457.89
- forearm  1     33.75 2435.8 458.19
- age      1     35.00 2437.0 458.28
- chest    1     42.28 2444.3 458.77
- BMI      1     59.35 2461.4 459.92
- hip      1    115.34 2517.4 463.63
- wrist    1    163.81 2565.9 466.78
- abdomen  1    944.72 3346.8 510.62
```

step() calls **add1()** and **drop1()** as long as the AIC cannot be reduced further.

– *Comparison of the models with* **anova()**

```
> anova(model.lmstep, model.lm1,model.lm)

Analysis of Variance Table

Model 1: body.fat ~ age + BMI + chest + abdomen + hip + forearm + wrist
Model 2: body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + ankle + bicep + forearm + wrist
Model 3: body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    157 2402.0
2    151 2346.2  6    55.857 0.5952 0.7338
3    150 2346.2  1     0.032 0.0021 0.9638
```

By using the smaller model model.lmstep, no essential information is lost, therefore it can be used for the prediction instead of model.lm.

```
> pred.lmstep <- predict(model.lmstep,newdata = fat[test,])
> cor(fat[test,"body.fat"],pred.lmstep)^2 # R^2 for test data

[1] 0.7372035

> mean((fat[test,"body.fat"]-pred.lmstep)^2) # MSE_test

[1] 16.87865
```

In this case, however, we seem to have almost the same performance as for the full model.

4.2.5 Best subset regression with Leaps and Bound algorithm

```
> library(leaps)
> lm.regsubset <- regsubsets(body.fat~., data=fat, nbest = 1, subset=train)
> summary(lm.regsubset)

Subset selection object
Call: regsubsets.formula(body.fat ~ ., data = fat, nbest = 1, subset = train)
14 Variables  (and intercept)
        Forced in Forced out
age         FALSE      FALSE
weight      FALSE      FALSE
height      FALSE      FALSE
BMI         FALSE      FALSE
neck        FALSE      FALSE
```

```
chest       FALSE      FALSE
abdomen     FALSE      FALSE
hip         FALSE      FALSE
thigh       FALSE      FALSE
knee        FALSE      FALSE
ankle       FALSE      FALSE
bicep       FALSE      FALSE
forearm     FALSE      FALSE
wrist       FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         age weight height BMI neck chest abdomen hip thigh knee ankle bicep
1  ( 1 ) " " " "    " "    " " " "  " "   "*"     " " " "   " "  " "   " "
2  ( 1 ) " " "*"    " "    " " " "  " "   "*"     " " " "   " "  " "   " "
3  ( 1 ) " " " "    " "    " " " "  " "   "*"     "*" " "   " "  " "   " "
4  ( 1 ) " " " "    "*"    " " " "  " "   "*"     "*" " "   " "  " "   " "
5  ( 1 ) " " " "    "*"    " " " "  " "   "*"     "*" " "   " "  " "   " "
6  ( 1 ) "*" " "    "*"    " " " "  " "   "*"     "*" " "   " "  " "   " "
7  ( 1 ) "*" " "    "*"    " " " "  " "   "*"     "*" "*"   " "  " "   " "
8  ( 1 ) "*" " "    "*"    " " " "  "*"   "*"     "*" "*"   " "  " "   " "
         forearm wrist
1  ( 1 ) " "     " "
2  ( 1 ) " "     " "
3  ( 1 ) " "     "*"
4  ( 1 ) " "     "*"
5  ( 1 ) "*"     "*"
6  ( 1 ) "*"     "*"
7  ( 1 ) "*"     "*"
8  ( 1 ) "*"     "*"
```

regsubsets() in `library(leaps)` provides the “best” model for different sizes of subsets. Here only one “best” model per subset size was considered. The ranking of the models is done using the BIC measure.

```
> plot(lm.regsubset)
```

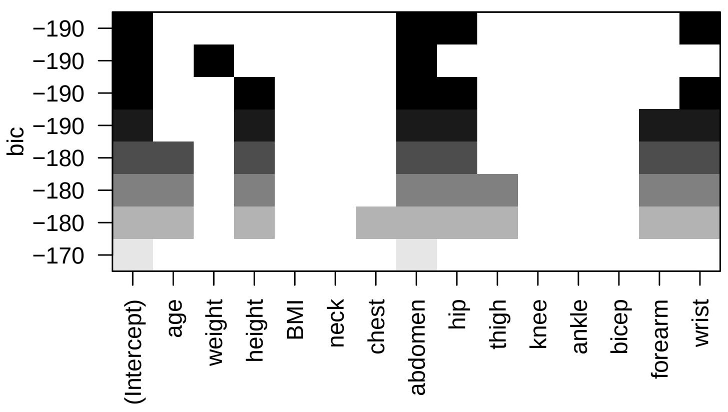

Figure 4.3: Model selection with **leaps()**

This plot shows the resulting models and their BIC value, coded in grey scale. The BIC does not improve after the fifth stage (starting from the bottom, see Figure 4.3). The optimal model can then be chosen from the models with “saturated” grey, and preferably that model is taken with the smallest number of variables.

A more intuitive plot might be that in Figure 4.4, where the BIC values are directly show. On the horizontal axis is the model size. The optimal model is the 2-variable model, with “weight” and “abdomen”.

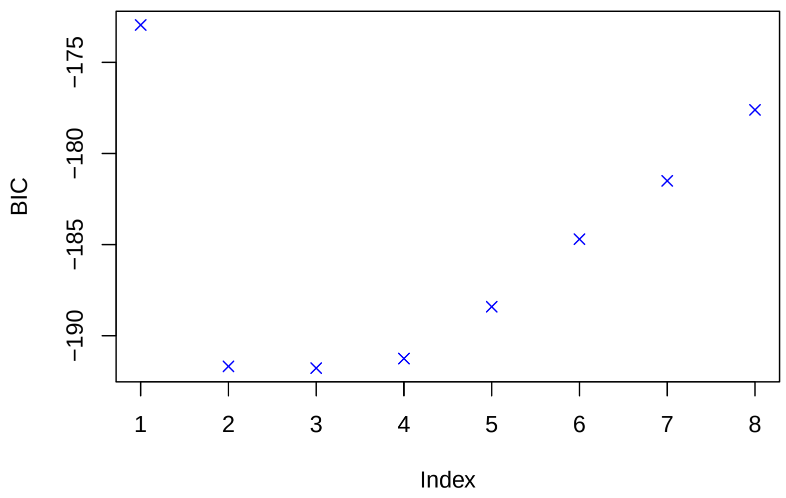

Figure 4.4: Model selection with **leaps**()

```
> modregsubset.lm <- lm(body.fat~weight+abdomen,data=fat,subset=train)
> pred.regsubset <- predict(modregsubset.lm,newdata = fat[test,])
> cor(fat[test,"body.fat"],pred.regsubset)^2 # R^2 for test data

[1] 0.7371287

> mean((fat[test,"body.fat"]-pred.regsubset)^2) # MSE_test

[1] 16.92699
```

Here again, the quality of the model does not quite change.

4.3 Methods using derived inputs as regressors in R

4.3.1 The problem of correlated regressors

The problem that occurs using correlated regressors is demonstrated using the following regression model:

$$y = X + \varepsilon, \quad \varepsilon \sim N(0, 0.25)$$

with the regressors

$$x_1 \sim U(0,1)$$
$$x_2 \sim U(0,1)$$
$$x_3 = x_1 + \nu_3, \quad \nu_3 \sim U(0, 0.1)$$

```
> x1 = runif(100)
> y = x1 + 0.5 * rnorm(100)
> summary(lm(y ~ x1))

Call:
lm(formula = y ~ x1)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.14872 -0.33275 -0.01504  0.31066  1.32760 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  -0.2633     0.1219  -2.160   0.0332 *
```

```
x1           1.3987     0.2048   6.831 7.15e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5292 on 98 degrees of freedom
Multiple R-squared:  0.3226,	Adjusted R-squared:  0.3157
F-statistic: 46.67 on 1 and 98 DF,  p-value: 7.147e-10

> x2 = runif(100)
> summary(lm(y ~ x1 + x2))

Call:
lm(formula = y ~ x1 + x2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.16328 -0.31706 -0.00776  0.30290  1.32335

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.22602    0.16554  -1.365    0.175
x1           1.38593    0.20922   6.624 1.95e-09 ***
x2          -0.06155    0.18402  -0.334    0.739
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5316 on 97 degrees of freedom
Multiple R-squared:  0.3234,	Adjusted R-squared:  0.3094
F-statistic: 23.18 on 2 and 97 DF,  p-value: 5.918e-09
```

- *Adding a highly correlated variable*

```
> x3 = x1 + 0.1 * runif(100)
> summary(lm(y ~ x1 + x3))

Call:
lm(formula = y ~ x1 + x3)

Residuals:
     Min       1Q   Median       3Q      Max
-1.14745 -0.31870 -0.01526  0.31558  1.33702

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1730     0.1526  -1.134   0.2596
x1            3.1738     1.8166   1.747   0.0838 .
x3           -1.7631     1.7929  -0.983   0.3279
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5293 on 97 degrees of freedom
Multiple R-squared:  0.3293,	Adjusted R-squared:  0.3154
F-statistic: 23.81 on 2 and 97 DF,  p-value: 3.868e-09
```

The fit of the model (R squared) does not change when x_3 is added, but now none of the coefficients is significant. To estimate the amount of collinearity, one could look at the correlation between the x-variables. Another option is the function `alias()`, which identifies exact linear dependencies.

```
> alias(lm(y ~ x1 + x3))

Model :
y ~ x1 + x3
```

Here, there are no exact linear dependencies, but we can create some to test the method:

```
> alias(lm(body.fat~., data=fat))

Model :
body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist

> fat.mod <- fat
> fat.mod$nonsense <- fat$neck+2*fat$chest-3*fat$abdomen
> alias(lm(body.fat~., data=fat.mod))

Model :
body.fat ~ age + weight + height + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist + nonsense

Complete :
         (Intercept) age weight height BMI neck chest abdomen hip thigh knee
nonsense  0           0   0      0      0   1    2     -3      0   0     0
         ankle bicep forearm wrist
nonsense  0     0     0       0
```

4.3.2 PCR

```
> library(pls)
> model.pcr <- pcr(body.fat~., data=fat, scale=TRUE, subset=train,
+                  validation="CV", segments=10, segment.type="random")
> summary(model.pcr)

Data:         X dimension: 165 14
        Y dimension: 165 1
Fit method: svdpc
Number of components considered: 14

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           7.508    5.893    4.896    4.854    4.636    4.612    4.502
adjCV        7.508    5.888    4.890    4.852    4.626    4.602    4.497
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       4.435    4.402    4.444     4.474     4.529     4.083     4.113
adjCV    4.417    4.389    4.430     4.458     4.512     4.065     4.095
       14 comps
CV        4.141
adjCV     4.121

TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X           65.45    76.56    82.56    87.26    90.43    92.92    95.17
body.fat    39.77    58.07    59.44    63.13    64.67    67.48    68.48
          8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X           96.75    97.96     98.84     99.38     99.75     99.96    100.00
body.fat    68.51    68.56     68.77     68.79     74.43     74.44     74.47
```

Note that the explanatory variables need to be scaled, therefore the option `scale=TRUE`. The plot (see Figure 4.5 left) suggests to use 12 components. The resulting predictions for the training data are shown in Figure 4.5 (right).

The predictions from the test data and some evaluation measure are shown below.

```
> pred.pcr <- predict(model.pcr,newdata=fat[test,],ncomp=12)
> cor(fat[test,"body.fat"],pred.pcr)^2 # R^2 for test data

[1] 0.7385471

> mean((fat[test,"body.fat"]-pred.pcr)^2) # MSE_test

[1] 16.81261
```

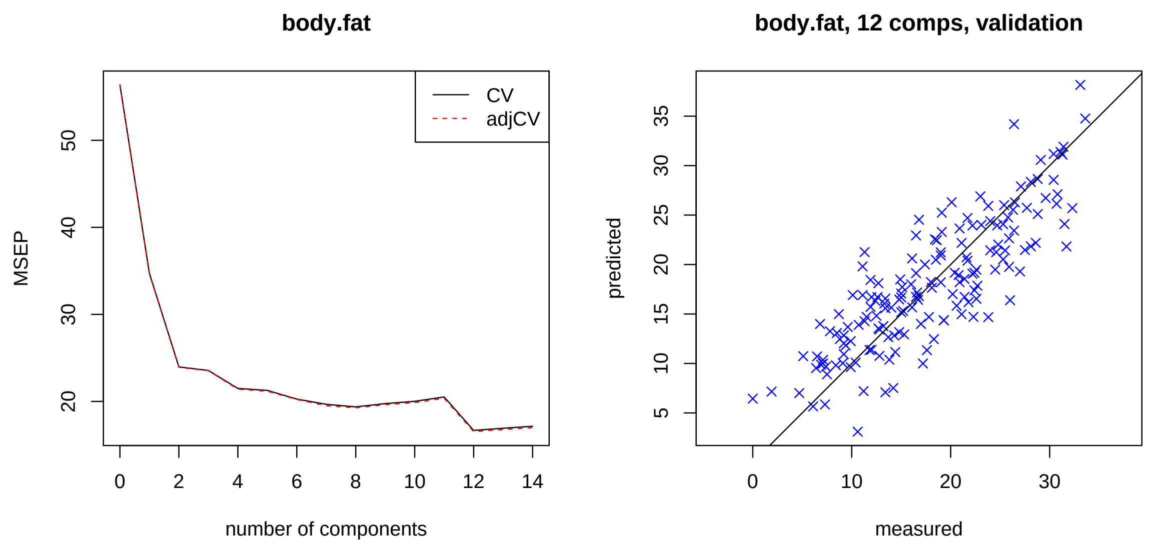

Figure 4.5: Validation plot, and measured versus predicted response for PCR.

4.3.3 PLS regression

```
> library(pls)
> model.pls <- plsr(body.fat~., data=fat, scale=TRUE, subset=train,
+                  validation="CV", segments=10, segment.type="random")
> summary(model.pls)

Data:         X dimension: 165 14
        Y dimension: 165 1
Fit method: kernelpls
Number of components considered: 14

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           7.508    5.491    4.553    4.391    4.258    4.198    4.124
adjCV        7.508    5.489    4.547    4.375    4.246    4.182    4.102
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       4.091    4.073    4.077     4.074     4.091     4.112     4.120
adjCV    4.073    4.057    4.061     4.059     4.075     4.094     4.101
       14 comps
CV        4.121
adjCV     4.102

TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X           64.68    75.99    80.62    84.30    87.41    90.23    92.93
body.fat    46.95    65.13    69.68    71.43    73.17    74.29    74.41
          8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X           94.35    96.27     97.62     98.28     99.25     99.72    100.00
body.fat    74.45    74.45     74.45     74.46     74.46     74.47     74.47
```

Again, the explanatory variables need to be scaled, therefore the option `scale=TRUE`. The plot (see Figure 4.6 left) suggests to use something like 7 components. The resulting predictions for the training data are shown in Figure 4.6 (right).

The predictions from the test data and some evaluation measure are shown below.

```
> pred.pls <- predict(model.pls,newdata=fat[test,],ncomp=7)
> cor(fat[test,"body.fat"],pred.pls)^2 # R^2 for test data

[1] 0.7395234

> mean((fat[test,"body.fat"]-pred.pls)^2) # MSE_test
```

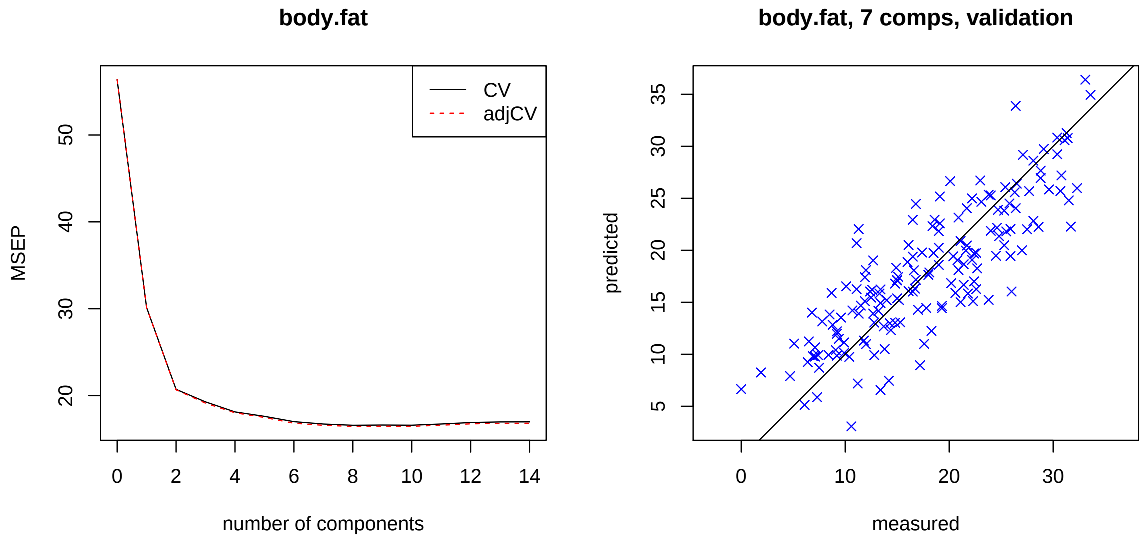

Figure 4.6: Validation plot, and measured versus predicted response for PLS.

```
[1] 16.74858
```

4.4 Shrinkage methods in R

4.4.1 Ridge regression

For a given grid of possible values of λ, the optimal tuning parameter has to be selected. Within the function `lm.ridge()`, generalized cross-validation (GCV) is used. This is a good approximation of leave-one-out (LOO) cross-validation by linear fits with quadratic errors. For a linear fit we have $\hat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{y}$ with the corresponding transformation matrix $\boldsymbol{S}$. In classical linear regression $\boldsymbol{S}$ is the hat matrix. One can show that the following equation holds:

$$\frac{1}{n}\sum_{i=1}^{n}\left[y_i - \hat{f}^{-i}(\boldsymbol{x}_i)\right]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - S_{ii}}\right]^2$$

Here, $\hat{f}^{-i}(\boldsymbol{x}_i)$ is a fit without observation i, $\hat{f}(\boldsymbol{x}_i)$ is a fit to all data, and S_{ii} is the i-th diagonal element of $\boldsymbol{S}$. The GCV approximation is

$$\text{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - trace(\boldsymbol{S})/n}\right]^2,$$

where $trace(\boldsymbol{S}) = \sum_{i=1}^{n} S_{ii}$. The idea is thus to replace S_{ii} by an average value, and since $trace(\boldsymbol{S})$ is the effective number of parameter, one has a computational advantage if this trace is easier to compute than the individual elements S_{ii}.

Now the model is fit to the training body fat data, and the resulting GCV errors are shown in Figure 4.7.

```
> library(MASS)
> model.ridge <- lm.ridge(body.fat~., data=fat, lambda=seq(0,15, by=0.2), subset=train)
> plot(model.ridge$lambda,model.ridge$GCV,type="l")
```

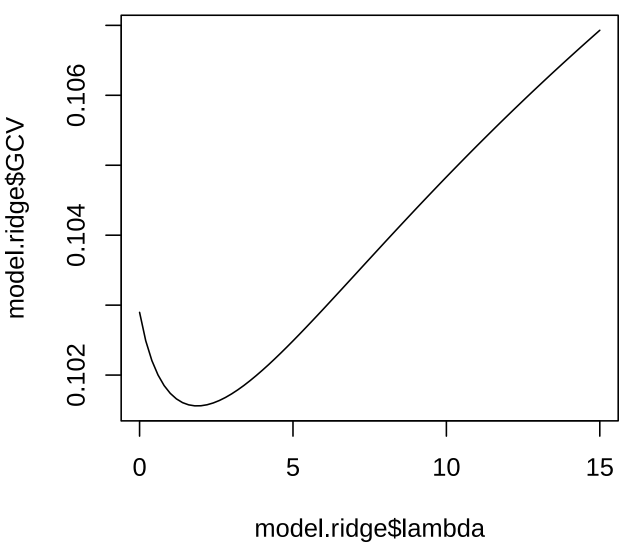

Figure 4.7: Optimal selection of the tuning parameter λ with GCV.

Figure 4.7 shows a very clear minimum at a specific value of λ. This value can be identified with the `select()` function.

```
> library(MASS)
> select(model.ridge)

modified HKB estimator is 2.168309
modified L-W estimator is 4.525899
smallest value of GCV  at 1.8

> lambda.opt <- model.ridge$lambda[which.min(model.ridge$GCV)]
```

This gives now the optimal tuning parameter, and some other characteristics which have been proposed for tuning parameter selection (HBK = Hoerl-Kennard coefficient, LW = Lawless-Wang coefficient) are returned as well.

One can also look at the whole solution path for each coefficient, depending on the values of the tuning parameter λ. This is shown in Figure 4.8, where each line corresponds to the resulting coefficients for one specific variable.

With the optimal values of λ, we want to predict the outcome variable for the test data. Let us first again compute the model for the optimized λ.

```
> # Prediction with Ridge:
> mod.ridge <- lm.ridge(body.fat~., data=fat, lambda = lambda.opt, subset=train)
```

Be careful here: The resulting coefficients are for **scaled** x-variables. For the prediction on unscaled data we need to back-transform them to the original scale. This is done with `coef()`, applied on the result object, where the scales are still stored, as well as the mean of the response, which is the intercept:

```
> mod.ridge$coef # coefficients for scaled x

        age      weight      height         BMI        neck       chest
 0.78235616 -0.07457382 -0.34385066  1.03933922 -0.30950292 -0.63353943
    abdomen         hip       thigh        knee       ankle       bicep
```

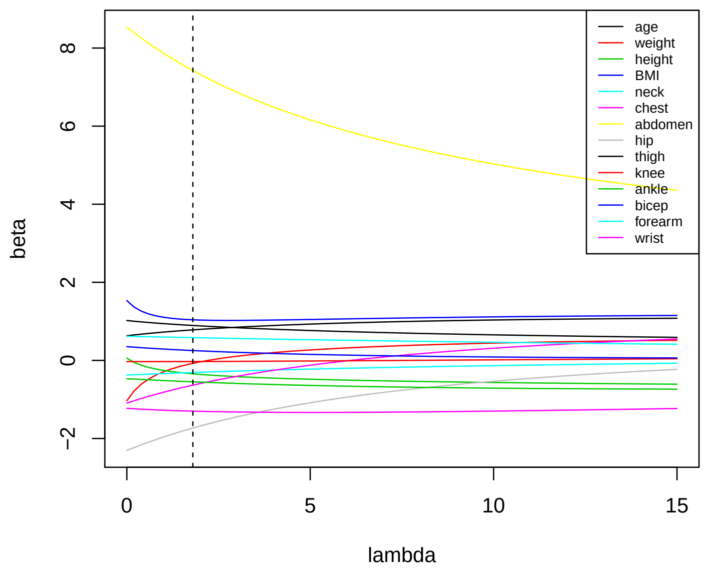

Figure 4.8: Ridge coefficients with varying λ. The optimal tuning parameter is indicated with the dashed line.

```
 7.42343629 -1.73224089  0.89443532 -0.02883642 -0.54814379  0.25192251
    forearm       wrist
 0.58262075 -1.29940964
> ridge.coef <- coef(mod.ridge)
> ridge.coef # coefficients in original scale + intercept
                     age       weight       height          BMI         neck
 3.367460057  0.061373666 -0.002680338 -0.135686607  0.314311555 -0.132886946
       chest      abdomen          hip        thigh         knee        ankle
-0.077906397  0.730676903 -0.259680747  0.173865322 -0.012053075 -0.396166795
       bicep      forearm        wrist
 0.083260870  0.278848086 -1.407762749
```

Now we predict the outcome variable for the test data. It turns out that the performance of the model is quite competitive.

```
> pred.ridge <- as.matrix(cbind(rep(1,length(test)),fat[test,-1]))%*%ridge.coef
> # plot(fat[test,"body.fat"],pred.ridge)
> # abline(c(0,1))
> cor(fat[test,"body.fat"],pred.ridge)^2 # R^2 for test data
         [,1]
[1,] 0.740998
> mean((fat[test,"body.fat"]-pred.ridge)^2) # MSE_test
[1] 16.78654
```

4.4.2 Lasso regression

We recommend to use the implementation in the package `glmnet`, which is very flexible. One could also do Ridge regression with this package, or extend Lasso regression to the ENET (Elastic Net), combining the L_1 and L_2 penalties.

Let us fit a Lasso model for the training data of body fat. The whole solution path for different values of the L_1 norm is computed, and it is visualized in Figure 4.9.

```
> library(glmnet)
> res <- glmnet(as.matrix(fat[train,-1]),fat[train,1])
> # print(res)
> plot(res)
```

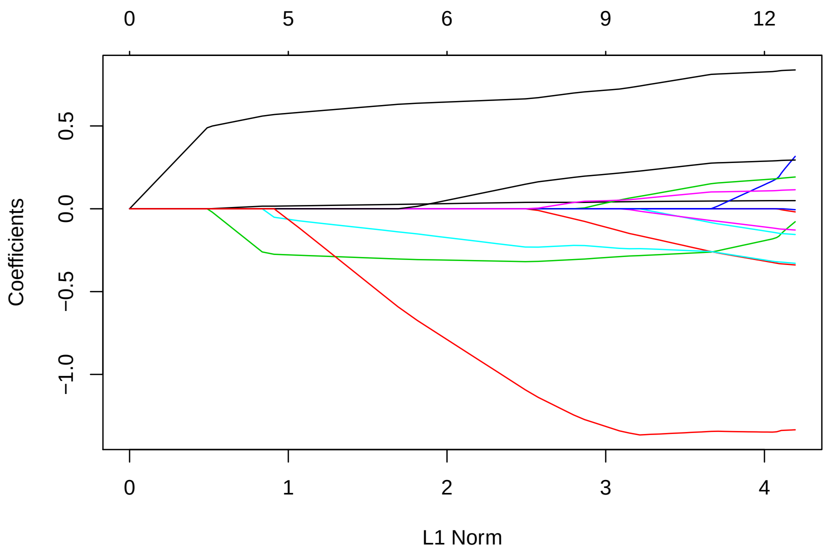

Figure 4.9: Lasso regression coefficients for varying values of the size of the L_1 norm. On top we can see the number of variables in the model.

To identify the appropriate tuning parameter, we run a cross-validation routine and plot the result, see Figure 4.10.

```
> res.cv <- cv.glmnet(as.matrix(fat[train,-1]),fat[train,1])
```

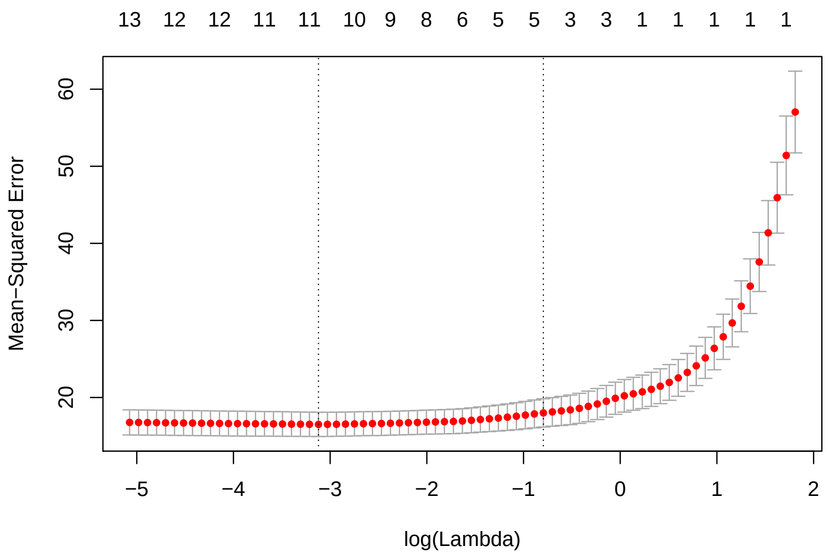

Figure 4.10: Cross-validated MSE for different values of λ in Lasso regression.

Figure 4.10 shows the MSE together with their standard errors. The left dashed line indicates the smallest MSE, and the right dashed line points at the optimal λ for with the MSE is still below the bound defined by the smallest MSE plus its standard error. This λ is selected if we go for the "one-standard error rule" (default). The resulting coefficients are:

```
> coef(res.cv,s="lambda.1se")

15 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -12.56008448
age           0.01790817
weight        .
height       -0.28074139
BMI           .
neck          .
chest         .
abdomen       0.58165319
hip           .
thigh         .
knee          .
ankle        -0.07271589
bicep         .
forearm       .
wrist        -0.11538100
```

We can see that the resulting coefficient vector is sparse, we obtain several zero entries and thus variable selection.

Finally, with the optimized model we predict the test data, and derive some evaluation characteristis. The model is competitive.

```
> pred.lasso <- predict(res.cv,newx=as.matrix(fat[test,-1]),s="lambda.1se")
> cor(fat[test,"body.fat"],pred.lasso)^2 # R^2 for test data

             1
[1,] 0.7299078

> mean((fat[test,"body.fat"]-pred.lasso)^2) # MSE_test

[1] 18.58005

> # plot(fat[test,"body.fat"],pred.lasso)
> # abline(c(0,1))
```

The resulting predictions from Ridge and Lasso regression are compared in Figure 4.11.

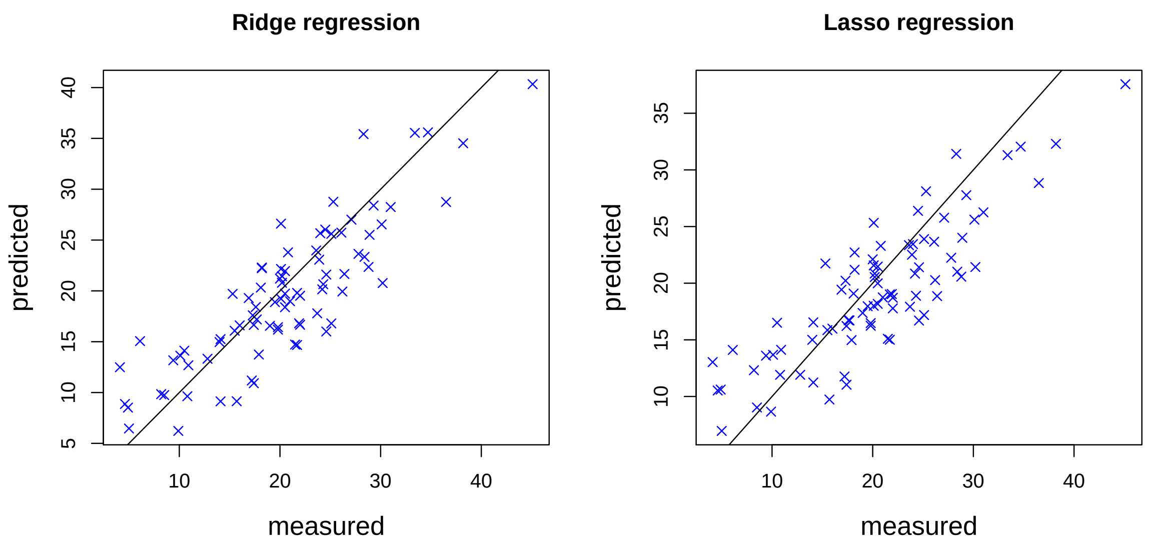

Figure 4.11: Measured versus predicted response for the test data, for Ridge (left) and LAsso (right) regression.

Chapter 5

Linear methods for classification

Linear classification tries to find linear functions of the form (1.1) which separate the observations into different classes. Observations within one group should have as many similar features as possible whereas observations of different groups should have little in common. If the predictor $G(\boldsymbol{x})$ takes on values in a discrete set $\mathcal{G}$, we can always divide the input space into different regions corresponding to the classification. The decision boundaries can either be smooth or rough. Assuming that we have K groups and the fitted linear model for the kth regressor variable is $\hat{f}_k(\boldsymbol{x}) = \hat{\beta}_{k0} + \hat{\boldsymbol{\beta}}_k^\top \boldsymbol{x}$. Then the decision boundary between class k and l is the set of points for which $\hat{f}_k(\boldsymbol{x}) = \hat{f}_l(\boldsymbol{x})$, that is, $\{\boldsymbol{x} : (\hat{\beta}_{k0} - \hat{\beta}_{l0}) + (\hat{\boldsymbol{\beta}}_k - \hat{\boldsymbol{\beta}}_l)^\top \boldsymbol{x} = 0\}$. New data points $\boldsymbol{x}$ can then be classified with this predictor.

5.1 Linear regression of an indicator matrix

Here each of the response categories is coded by an indicator variable. Thus if $\mathcal{G}$ has K classes, there will be K such indicators

$$y_k \quad \text{with } k = 1, 2, \ldots, K$$

with

$$y_k = \begin{cases} 1 & \text{for } G = k \\ 0 & \text{otherwise} \end{cases}$$

These response variables are collected in a vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_K)$, and the n training instances form an $(n \times K)$ indicator response matrix $\boldsymbol{Y}$, with 0/1 values as indicators for the class membership. We fit a linear regression model to each of the columns $\boldsymbol{y}_k$ of $\boldsymbol{Y}$ which is given by

$$\hat{\boldsymbol{\beta}}_k = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}_k \quad \text{with } k = 1, 2, \ldots, K$$

The $\hat{\boldsymbol{\beta}}_k$ can be combined in a $((p+1) \times K)$ matrix $\hat{\boldsymbol{B}}$ since

$$\begin{aligned} \hat{\boldsymbol{B}} &= (\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, \ldots, \hat{\boldsymbol{\beta}}_K) \\ &= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K) \\ &= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \end{aligned}$$

The estimated values are then

$$\begin{aligned} \hat{\boldsymbol{Y}} &= \boldsymbol{X} \hat{\boldsymbol{B}} \\ &= \boldsymbol{X} (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y} \end{aligned}$$

A new observation $\boldsymbol{x}$ can be classified as follows:

- Compute the K vector of the fitted values

$$\hat{f}(\boldsymbol{x}) = \left[(1, \boldsymbol{x}^\top)\hat{\boldsymbol{B}}\right]^\top = \left[\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_K(\boldsymbol{x})\right]^\top$$

- identify the largest component $\hat{f}(\boldsymbol{x})$ and classify $\boldsymbol{x}$ accordingly:

$$\hat{G}(\boldsymbol{x}) = \underset{k \in \mathcal{G}}{\operatorname{argmax}} \hat{f}_k(\boldsymbol{x})$$

Section 6.1, page 47

5.2 Linear discriminant analysis (LDA)

5.2.1 Classical LDA

$\mathcal{G}$ consists of K classes. Let the probability of an observation belonging to class k be (the prior probability) $\pi_k,\ k = 1, 2, \ldots, K$ with $\sum_{k=1}^K \pi_k = 1$. Suppose $h_k(\boldsymbol{x})$ is the density function of $\boldsymbol{x}$ in class $G = k$. Then

$$P(G = k|\boldsymbol{x}) = \frac{h_k(\boldsymbol{x})\pi_k}{\sum\limits_{l=1}^{K} h_l(\boldsymbol{x})\pi_l}$$

is the conditional probability that with a given observation $\boldsymbol{x}$ the random variable G is k. $h_k(\boldsymbol{x})$ is often assumed to be the density of a multivariate normal distribution φ_k

$$\varphi_k(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)}{2}\right\}$$

LDA arises in the special case when we assume that the classes have a common covariance $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma},\ k = 1, 2, \ldots, K$.

Comparing these two classes, it is sufficient to look at the log-ratio

$$\begin{aligned}
\log \frac{P(G = k|\boldsymbol{x})}{P(G = l|\boldsymbol{x})} &= \log \frac{\varphi_k(\boldsymbol{x})\pi_k}{\varphi_l(\boldsymbol{x})\pi_l} = \log \frac{\varphi_k(\boldsymbol{x})}{\varphi_l(\boldsymbol{x})} + \log \frac{\pi_k}{\pi_l} \\
&= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_l)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_l) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_l)
\end{aligned}$$

The decision boundary between the classes k and l is

$$P(G = k|\boldsymbol{x}) = P(G = l|\boldsymbol{x})$$

which is linear in $\boldsymbol{x}$ (in p dimensions this is a hyperplane).

From the log-ratio we get the **linear discriminant function:**

$$\begin{aligned}
\log \frac{P(G=k|\boldsymbol{x})}{P(G=l|\boldsymbol{x})} &= \log(1) = 0 = \log \frac{\varphi_k(\boldsymbol{x})\pi_k}{\varphi_l(\boldsymbol{x})\pi_l} \\
&= \log \varphi_k(\boldsymbol{x}) + \log \pi_k - \log \varphi_l(\boldsymbol{x}) - \log \pi_l \\
&= \log \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log \pi_k \\
&- \log \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_l)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_l) - \log \pi_l \\
&= -\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} \underbrace{+\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \pi_k}_{\delta_k(\boldsymbol{x})} \\
&\quad +\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} \underbrace{-\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l + \frac{1}{2}\boldsymbol{\mu}_l^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l - \log \pi_l}_{-\delta_l(\boldsymbol{x})}
\end{aligned}$$

The linear discriminant function

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

provides an equivalent description of the decision rule

$$G(\boldsymbol{x}) = \underset{k}{\operatorname{argmax}}\, \delta_k(\boldsymbol{x})$$

The observation $\boldsymbol{x}$ is classified to the group where $\delta_k(\boldsymbol{x}),\ k = 1, 2, \dots, K$ is the largest. $\boldsymbol{x}$ is classified to class k if

$$\delta_k(\boldsymbol{x}) > \delta_l(\boldsymbol{x}) \quad \Longleftrightarrow \quad \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_l) - \frac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_l)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_l) > \log \frac{\pi_l}{\pi_k}$$

The parameters of the distribution $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ as well as the prior probabilities π_k are usually unknown and need to be estimated from the training data:

$$\begin{aligned}
\hat{\pi}_k &= \frac{n_k}{n} \qquad \text{with } n_k \dots \text{ amount of observations in group } k \\
\hat{\boldsymbol{\mu}}_k &= \sum_{g_i=k} \frac{\boldsymbol{x}_i}{n_k} \\
\hat{\boldsymbol{\Sigma}} &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{g_i=k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top
\end{aligned}$$

g_i indicates the true group number of observation $\boldsymbol{x}_i$. Using the linear discriminant function we now can estimate the group membership of $\boldsymbol{x}_i$. This gives a value for $\hat{G}(\boldsymbol{x}_i)$, which can now be compared to g_i. The aim is correct classification of as many observations as possible. The misclassification rate provides the relative amount of incorrectly classified observations.

Section 6.2.1, page 48

5.2.2 Quadratic discriminant analysis (QDA)

Just as in LDA we assume a prior probability π_k for class k, $k = 1, 2, \ldots, K$ with $\sum_{k=1}^{K} \pi_k = 1$. The conditional probability that G takes on the value k is

$$P(G = k|\boldsymbol{x}) = \frac{\varphi_k(\boldsymbol{x})\pi_k}{\sum\limits_{l=1}^{K} \varphi_l(\boldsymbol{x})\pi_l}$$

(φ is the density of the multivariate normal distribution). QDA does not assume the covariance matrices to be equal, which complicates the formulas. After some calculation we obtain the quadratic discriminant function

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) + \log \pi_k.$$

Observation $\boldsymbol{x}$ is classified to that group which yields the maximal value of the discriminant function. $\boldsymbol{\Sigma}_k$, $\boldsymbol{\mu}_k$ and π_k can be estimated from the training data.

Comparison QDA and LDA

With LDA we need to estimate much less parameters as with QDA (each covariance matrix $\hat{\boldsymbol{\Sigma}}_k$). Both methods work surprisingly well, even if the classes are not normally distributed and even if the equality of the covariance matrices is not given. The reason is most likely that the data can only support simple decision boundaries such as linear or quadratic, and the estimates provided via Gaussian models are stable.

Section 6.2.2, page 51

5.2.3 Regularized discriminant analysis

This method is a compromise between linear and quadratic discriminant analysis, that allows to shrink the separate covariances of QDA towards a common covariance as in LDA. These common (pooled) covariance matrices have the form

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{\sum_{k=1}^{K} n_k} \left(\sum_{k=1}^{K} n_k \hat{\boldsymbol{\Sigma}}_{\boldsymbol{k}} \right)$$

with n_k as the number of observations per group. With the pooled covariance matrices the regularized covariance matrices have the form

$$\hat{\boldsymbol{\Sigma}}_k(\alpha) = \alpha \hat{\boldsymbol{\Sigma}}_k + (1 - \alpha)\hat{\boldsymbol{\Sigma}}$$

with $\alpha \in [0, 1]$. α provides a compromise between LDA ($\alpha = 0$) and QDA ($\alpha = 1$). The idea is to keep the degrees of freedom flexible. α can be estimated using cross validation.

Section 6.2.3, page 51

5.3 Logistic regression

Logistic regression also deals with the problem of classifying observations that originate from 2 or more groups. The difference to the previous classification methods is that the output of logistic regression includes an inference statistic which provides information about which variables are well suitable for separating the groups, and which provide no contribution to this goal.

In logistic regression the posterior probabilities of the K classes are modeled by linear functions in $\boldsymbol{x}$, with the constraint that the probabilities remain in the interval $[0,1]$ and that they sum up to 1. Let us consider the following models:

$$\begin{aligned} \log \frac{P(G=1|\boldsymbol{x})}{P(G=K|\boldsymbol{x})} &= \beta_{10} + \boldsymbol{\beta}_1^\top \boldsymbol{x} \\ \log \frac{P(G=2|\boldsymbol{x})}{P(G=K|\boldsymbol{x})} &= \beta_{20} + \boldsymbol{\beta}_2^\top \boldsymbol{x} \\ &\vdots \\ \log \frac{P(G=K-1|\boldsymbol{x})}{P(G=K|\boldsymbol{x})} &= \beta_{(K-1)0} + \boldsymbol{\beta}_{K-1}^\top \boldsymbol{x} \end{aligned}$$

Here we chose class K to be the denominator, but the choice of the denominator is arbitrary in the sense that the estimates are equivariant under that choice. After a few calculations we get

$$P(G=k|\boldsymbol{x}) = P(G=K|\boldsymbol{x}) \exp\left\{\beta_{k0} + \boldsymbol{\beta}_k^\top \boldsymbol{x}\right\} \quad \text{for } k = 1,2,\dots,K-1$$

$$\sum_{k=1}^{K} P(G=k|\boldsymbol{x}) = 1 = P(G=K|\boldsymbol{x}) \left[1 + \sum_{k=1}^{K-1} \exp\left\{\beta_{k0} + \boldsymbol{\beta}_k^\top \boldsymbol{x}\right\}\right]$$

$$\begin{aligned} P(G=K|\boldsymbol{x}) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp\left\{\beta_{l0} + \boldsymbol{\beta}_l^\top \boldsymbol{x}\right\}} \\ P(G=k|\boldsymbol{x}) &= \frac{\exp\left\{\beta_{k0} + \boldsymbol{\beta}_k^\top \boldsymbol{x}\right\}}{1 + \sum_{l=1}^{K-1} \exp\left\{\beta_{l0} + \boldsymbol{\beta}_l^\top \boldsymbol{x}\right\}} \quad \text{for } k = 1,2,\dots,K-1 \end{aligned}$$

Special case of $K = 2$ classes: In this case, the model is

$$\log \frac{P(G=1|\boldsymbol{x})}{P(G=2|\boldsymbol{x})} = \beta_0 + \boldsymbol{\beta}_1^\top \boldsymbol{x}$$

or

$$\begin{aligned} P(G=1|\boldsymbol{x}) &= \frac{\exp\{\beta_0 + \boldsymbol{\beta}_1^\top \boldsymbol{x}\}}{1 + \exp\{\beta_0 + \boldsymbol{\beta}_1^\top \boldsymbol{x}\}} \\ P(G=2|\boldsymbol{x}) &= \frac{1}{1 + \exp\{\beta_0 + \boldsymbol{\beta}_1^\top \boldsymbol{x}\}} \\ \underbrace{P(G=1|\boldsymbol{x})}_{p_1(\boldsymbol{x})} + \underbrace{P(G=2|\boldsymbol{x})}_{p_2(\boldsymbol{x})} &= 1 \end{aligned}$$

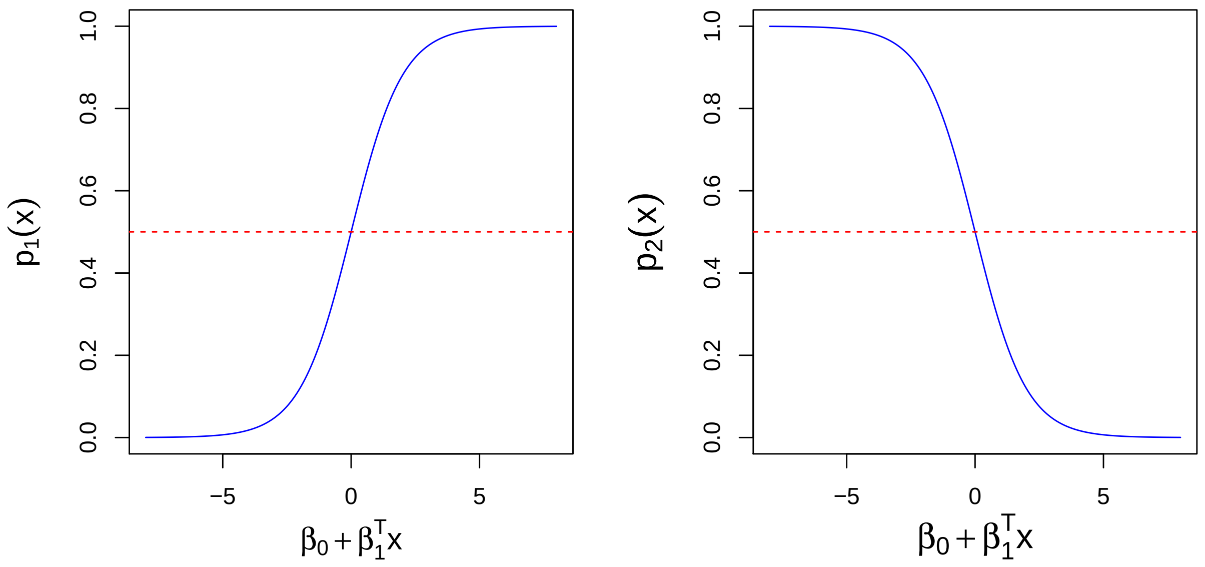

Figure 5.1: Logistic functions $p_1(\boldsymbol{x})$ (left) and $p_2(\boldsymbol{x})$ (right). Values above the dashed lines would be assigned to the corresponding groups.

Figure 5.1 visualizes the so-called *logistic functions*, i.e. the probabilities $p_1(\boldsymbol{x})$ (left) and $p_2(\boldsymbol{x})$ (right). A natural cut-off for the assignment of an observations to one of the groups would be 0.5 (dashed lines).

The parameters can be estimated using the ML method. The log-likelihood function is

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log p_{g_i}(\boldsymbol{x}_i; \boldsymbol{\beta}),$$

where

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \boldsymbol{\beta}_1 \end{pmatrix}$$

and $\boldsymbol{x}_i$ includes the intercept. p_{g_i} is the probability that observation $\boldsymbol{x}_i$ belongs to class $g_i = 1, 2$. With

$$p(\boldsymbol{x}; \boldsymbol{\beta}) = p_1(\boldsymbol{x}; \boldsymbol{\beta}) = 1 - p_2(\boldsymbol{x}; \boldsymbol{\beta})$$

and an indicator $y_i = 1$ for $g_i = 1$ and $y_i = 0$ for $g_i = 2$, $l(\boldsymbol{\beta})$ can be written as

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \{y_i \log p(\boldsymbol{x}_i; \boldsymbol{\beta}) + (1 - y_i) \log [1 - p(\boldsymbol{x}_i; \boldsymbol{\beta})]\} \\ &= \sum_{i=1}^{n} \left\{ y_i \log \left[\frac{\exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i)}{1 + \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i)} \right] + (1 - y_i) \log \left[1 - \frac{\exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i)}{1 + \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i)} \right] \right\} \\ &= \sum_{i=1}^{n} \left\{ y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i - y_i \log \left[1 + \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i) \right] - (1 - y_i) \log \left[1 + \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i \right] \right\} \\ &= \sum_{i=1}^{n} \left\{ y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i - \log \left[1 + \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i) \right] \right\} \end{aligned}$$

To maximize the log-likelihood we set its derivative equal to zero. These score equations are $p+1$ nonlinear equations in $\boldsymbol{\beta}$ of the form

$$\begin{aligned}\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{n} \left\{ y_i \boldsymbol{x}_i - \frac{1}{1+\exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i)} \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_i) \boldsymbol{x}_i \right\} \\ &= \sum_{i=1}^{n} \left[y_i - p(\boldsymbol{x}_i; \boldsymbol{\beta}) \right] \boldsymbol{x}_i = \boldsymbol{0}.\end{aligned}$$

To solve these equations, we use the *Newton-Raphson algorithm*, which requires the second derivative or Hessian matrix:

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} = \ldots = -\sum_{i=1}^{n} p(\boldsymbol{x}_i; \boldsymbol{\beta})[1 - p(\boldsymbol{x}_i; \boldsymbol{\beta})] \boldsymbol{x}_i \boldsymbol{x}_i^\top$$

Starting with $\boldsymbol{\beta}_{old}$ we get

$$\boldsymbol{\beta}_{new} = \boldsymbol{\beta}_{old} - \left(\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \right)^{-1} \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}},$$

where the derivatives are evaluated at $\boldsymbol{\beta}_{old}$.
Matrix notation provides a better insight in that method:

$\boldsymbol{y} \ldots (n \times 1)$ vector of the y_i
$\boldsymbol{X} \ldots (n \times (p+1))$ matrix of observations $\boldsymbol{x}_i$
$\boldsymbol{p} \ldots (n \times 1)$ vector of estimated probabilities $p(\boldsymbol{x}_i, \boldsymbol{\beta}_{old})$
$\boldsymbol{W} \ldots (n \times n)$ diagonal matrix with weights $p(\boldsymbol{x}_i, \boldsymbol{\beta}_{old})(1 - p(\boldsymbol{x}_i, \boldsymbol{\beta}_{old}))$ in the diagonal

and thus:

$$\begin{aligned}\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{p}) \\ \frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} &= -\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}\end{aligned}$$

The Newton-Raphson algorithm has the form

$$\begin{aligned}\boldsymbol{\beta}_{new} &= \boldsymbol{\beta}_{old} + (\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{p}) \\ &= (\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{W} (\boldsymbol{X} \boldsymbol{\beta}_{old} + \boldsymbol{W}^{-1} (\boldsymbol{y} - \boldsymbol{p})) \\ &= \underbrace{(\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{W}}_{\text{weighted LS}} \boldsymbol{z}\end{aligned}$$

with the adjusted response

$$\boldsymbol{z} = \boldsymbol{X} \boldsymbol{\beta}_{old} + \underbrace{\boldsymbol{W}^{-1} (\boldsymbol{y} - \boldsymbol{p})}_{\text{adjustment}}).$$

This algorithm is also referred to as IRLS (iteratively reweighted LS), since each iteration solves the weighted least squares problem

$$\boldsymbol{\beta}_{new} \longleftarrow \underset{\boldsymbol{\beta}}{\operatorname{argmin}} (\boldsymbol{z} - \boldsymbol{X} \boldsymbol{\beta})^\top \boldsymbol{W} (\boldsymbol{z} - \boldsymbol{X} \boldsymbol{\beta}).$$

Some remarks:

- $\boldsymbol{\beta} = \mathbf{0}$ is a good starting value for the iterative procedure, although convergence is never guaranteed.
- For $K \geq 3$ the Newton-Raphson algorithm can also be expressed as an IRLS algorithm, but in this case $\boldsymbol{W}$ is no longer a diagonal matrix.
- The logistic regression is mostly used for modeling and inference. The goal is to understand the role of the input variables in explaining the outcome.

Comparison of logistic regression and LDA

Logistic regression uses

$$\log \frac{P(G=k|\boldsymbol{x})}{P(G=q|\boldsymbol{x})} = \beta_{k0} + \boldsymbol{\beta}_k^\top \boldsymbol{x},$$

whereas LDA uses

$$\begin{aligned} \log \frac{P(G=k|\boldsymbol{x})}{P(G=K|\boldsymbol{x})} &= \log \frac{\pi_k}{\pi_K} - \frac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_K)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_K) \\ &= \alpha_{k0} - \boldsymbol{\alpha}_k^\top \boldsymbol{x} \end{aligned}$$

The linearity in $\boldsymbol{x}$ in LDA is achieved by:

- the assumption of equal covariance matrices
- the assumption of multivariate normally distributed groups

Even though the models have the same form, the coefficients are estimated differently. The joint distribution of $\boldsymbol{x}$ and G can be expressed as

$$P(\boldsymbol{x}, G=k) = P(\boldsymbol{x})P(G=k|\boldsymbol{x})$$

Logistic regression does not specify $P(\boldsymbol{x})$, LDA assumes a mixture model (with φ as a normal distribution):

$$P(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \varphi(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

Section 6.3, page 51

Chapter 6

Linear methods for classification in R

6.1 Linear regression of an indicator matrix in R

- *Classification with indicator matrix based on the "Pima Indian" data*

```
> library(mlbench)
> data(PimaIndiansDiabetes2)
> #plot(PimaIndiansDiabetes2)
> pid <- na.omit(PimaIndiansDiabetes2)
```

The data consist of a population of 392 women with "Pima Indian" heritage, who live in the area of Phoenix, Arizona.They were tested for diabetes. The goal is to get a classification rule for the diagnosis of diabetes. The variables are:

- `npreg`: number of pregnancies
- `glu`: plasma glucose concentration (glucose tolerance test)
- `bp`: diastolic blood pressure (mm Hg)
- `skin`: triceps skin thickness (mm)
- `bmi`: BMI
- `ped`: diabetes pedigree function
- `age`: age in years
- `type`: "pos" or "neg" for diabetes diagnosis

Generation of the indicator matrix for regression:

```
> pidind <- pid
> ind <- ifelse(pid$diabetes=="neg",0,1)
> pidind$diabetes <- cbind(1-ind,ind)
```

It would be sufficient to consider only one indicator variable since we have a symmetric problem.

Random selection (using a fixed random seed) of the training data, and fitting of a linear model:

```
> set.seed(101)
> train <- sample(1:nrow(pid), 300)
> mod.ind <- lm(diabetes~., data=pidind[train,])
> mod.ind
```

```
Call:
lm(formula = diabetes ~ ., data = pidind[train, ])

Coefficients:
                          ind
(Intercept)   2.059e+00  -1.059e+00
pregnant     -1.292e-02   1.292e-02
glucose      -6.046e-03   6.046e-03
pressure      3.797e-04  -3.797e-04
triceps      -3.315e-03   3.315e-03
insulin       6.719e-05  -6.719e-05
mass         -7.820e-03   7.820e-03
pedigree     -8.852e-02   8.852e-02
age          -7.654e-03   7.654e-03
```

Fitting a linear model to the data yields a regression coefficients $\hat{\boldsymbol{\beta}}_k$ for each $\boldsymbol{y}_k$.

The model is applied to the test data, and the resulting misclassification rate is computed.

```
> mod.pred <- predict(mod.ind, newdata=pidind[-train,])
> class.pred <- apply(mod.pred,1,which.max) # class prediction
> TAB <- table(pid$diabetes[-train], class.pred)
> mklrate<-1-sum(diag(TAB))/sum(TAB)
> mklrate
[1] 0.2391304
```

The resulting misclassification rate is 0.239.
The misclassification rates of different classification methods will be collected in a table and then compared and analyzed throughout the rest of this manuscript.

	INDR	LDA	QDA	RDA	GLM
MKR	0.239				

6.2 Linear Discriminant Analysis in R

6.2.1 Classical LDA

- *LDA for 2-dimensional multivariate normally distributed data with parameters* $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}$

```
> mu1<-c(0,0)
> mu2<-c(3.5,1)
> sig<-matrix(c(1.5,1,1,1.5),ncol=2)
```

The ellipses in Figure 6.1 are so-called tolerance ellipses which (in case of multivariate normal distribution) include the inner 95% of the data of each group. The LDA separation line was determined using the known population parameters $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$. The command **lda()** can be found in `library(MASS)`.

- *Change of the prior probabilities – see Figure 6.2*

When changing the prior probabilities, the LDA separation line is moved towards the group with smaller prior probability (see Figure 6.2). This is done because LDA minimizes the probability of misclassification.

- *Classification of the PimaIndianDiabetes data with* **lda()**

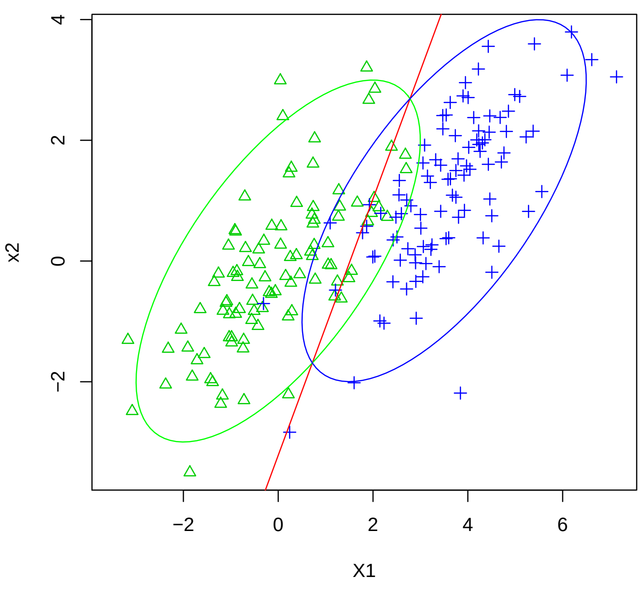

Figure 6.1: LDA separation line using the population parameters and $\pi_1 = \pi_2$

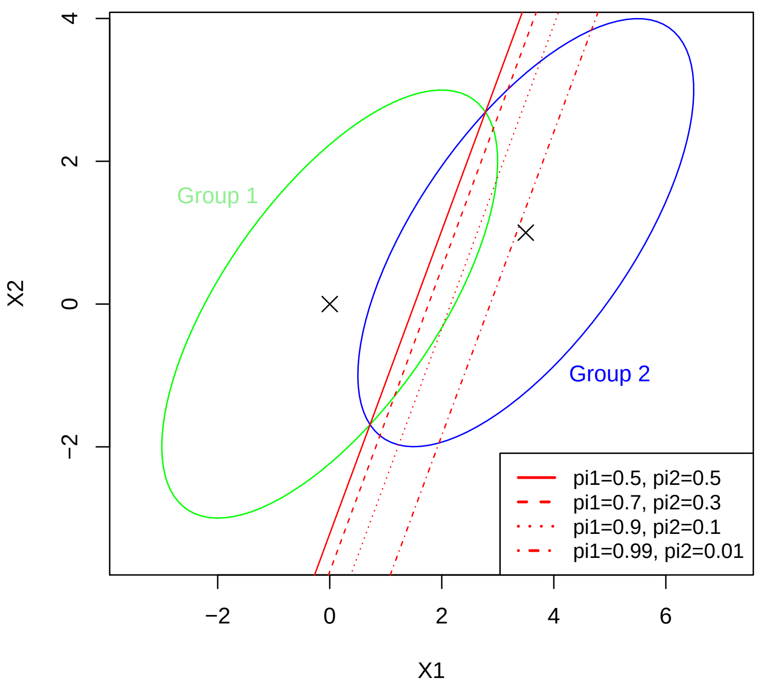

Figure 6.2: Change of the prior probabilities

An advanced form of an evaluation (we could think of an even more advanced form) is as follows:

```
> library(MASS)
> mypred <- function(object, newdata) UseMethod("mypred", object)
> mypred.lda <- function(object, newdata){
+               predict(object, newdata = newdata)$class
+ }
> library(ipred)
> CEE <-  control.errorest(k = 5, nboot=10)
> ldacvest <- errorest(diabetes~., data=pid[train,], model=lda, predict=mypred,
+          est.para=CEE)
> ldacvest

Call:
errorest.data.frame(formula = diabetes ~ ., data = pid[train,
    ], model = lda, predict = mypred, est.para = CEE)

         5-fold cross-validation estimator of misclassification error

Misclassification error:  0.2267

> ldabest <- errorest(diabetes~., data=pid[train,], model=lda,predict=mypred,
+          estimator="boot", est.para=CEE)
> ldabest

Call:
errorest.data.frame(formula = diabetes ~ ., data = pid[train,
    ], model = lda, predict = mypred, estimator = "boot", est.para = CEE)

         Bootstrap estimator of misclassification error
         with 10 bootstrap replications

Misclassification error:  0.2391
Standard deviation: 0.0124
```

The command **errorest()** can be found in `library(ipred)` and can be used to estimate the misclassification rate with cv or bootstrap.

A simple evaluation based on just one training and test data set can give quite unreliable results:

```
> set.seed(101)
> train <- sample(1:nrow(pid), 300)
> mod.lda <- lda(diabetes~.,data=pid[train,])
> TAB <- table(pid[-train,]$diabetes,mypred(mod.lda,pid[-train,]))
> mkrlda <- 1-sum(diag(TAB))/sum(TAB)
> mkrlda

[1] 0.2391304
```

	INDR	LDA	QDA	RDA	GLM
MKR	0.239	0.239			

6.2.2 QDA

- *Classification of the PimaIndianDiabetes data with* **qda()**

```
> set.seed(101)
> train <- sample(1:nrow(pid), 300)
> mod.qda <- qda(diabetes~.,data=pid[train,])
> predictqda <- predict(mod.qda, pid[-train,])
> TAB <- table(pid$diabetes[-train],predictqda$class)
> mkrqda <- 1-sum(diag(TAB))/sum(TAB)
> mkrqda

[1] 0.25
```

	INDR	LDA	QDA	RDA	GLM
MKR	0.239	0.239	0.25		

6.2.3 Regularized discriminant analysis

- *Classification of the PimaIndianDiabetes data with* **rda()**

```
> library(klaR)
> mod.rda <- rda(diabetes~.,data=pid[train,])
> predictrda <- predict(mod.rda, pid[-train,])
> TAB <- table(pid$diabetes[-train], predictrda$class)
> mkrrda <- 1-sum(diag(TAB))/sum(TAB)
> mkrrda

[1] 0.2391304
```

	INDR	LDA	QDA	RDA	GLM
MKR	0.239	0.239	0.25	0.239	

6.3 Logistic regression in R

- *Fit of a model with* **glm()**

  Logistic regression can be carried out with the function **glm()** under the model of the binomial family. Syntax and interpretation of the inference statistic are in analogy to **lm()**. The tests for the single coefficients are similar to those from the linear model, but here they are based on the asymptotic normality of the parameter estimates that are MLEs. The standard errors are based on the second derivative of the log-likelihood function at the maximum, which is an indication of how rapidly the function decreases as one moves away from the peak.

```
> set.seed(101)
> train <- sample(1:nrow(pid), 300)
> modelglm <- glm(diabetes~.,data=pid, family=binomial, subset=train)
> summary(modelglm)

Call:
glm(formula = diabetes ~ ., family = binomial, data = pid, subset = train)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4400  -0.6729  -0.3530   0.6820   2.4859

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.7862578  1.3765057  -7.109 1.16e-12 ***
pregnant     0.0813129  0.0637407   1.276   0.2021
glucose      0.0370238  0.0066143   5.598 2.17e-08 ***
pressure    -0.0045533  0.0135442  -0.336   0.7367
triceps      0.0219238  0.0191473   1.145   0.2522
insulin     -0.0005762  0.0014721  -0.391   0.6955
mass         0.0608347  0.0300131   2.027   0.0427 *
pedigree     0.5898194  0.4901525   1.203   0.2288
age          0.0444517  0.0209460   2.122   0.0338 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 380.51  on 299  degrees of freedom
Residual deviance: 262.09  on 291  degrees of freedom
AIC: 280.09

Number of Fisher Scoring iterations: 5
```

The deviances are the negative contributions of each observation to the log-likelihood function. The “Null deviance” refers to the empty model, the “Residual deviance” to the full model.

- *Model selection with* **step()**: This is done in the same spirit as for the linear model.

```
> mod.glm <- step(modelglm,direction="both")

Start:  AIC=280.09
diabetes ~ pregnant + glucose + pressure + triceps + insulin +
    mass + pedigree + age

           Df Deviance    AIC
- pressure  1   262.20 278.20
- insulin   1   262.25 278.25
- triceps   1   263.41 279.41
- pedigree  1   263.61 279.61
- pregnant  1   263.73 279.73
<none>          262.09 280.09
- mass      1   266.27 282.27
- age       1   266.91 282.91
- glucose   1   299.22 315.22

                     ⋮

Step:  AIC=274.91
diabetes ~ glucose + mass + age

           Df Deviance    AIC
<none>          266.91 274.91
+ triceps   1   265.34 275.34
+ pregnant  1   265.44 275.44
+ pedigree  1   265.50 275.50
+ insulin   1   266.70 276.70
+ pressure  1   266.78 276.78
- mass      1   280.12 286.12
- age       1   285.74 291.74
- glucose   1   315.97 321.97
```

The result after stepwise logistic regression is a smaller model that should be able to have a better prediction performance than the full model. The inference statistic tells which variables are significantly contributing to the group separation:

```
> summary(mod.glm)

Call:
glm(formula = diabetes ~ glucose + mass + age, family = binomial,
    data = pid, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6686  -0.6674  -0.3642   0.6503   2.5019

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.931492   1.205761  -8.237  < 2e-16 ***
glucose      0.035407   0.005616   6.304 2.90e-10 ***
mass         0.078319   0.022551   3.473 0.000515 ***
age          0.063726   0.015525   4.105 4.05e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 380.51  on 299  degrees of freedom
Residual deviance: 266.91  on 296  degrees of freedom
AIC: 274.91

Number of Fisher Scoring iterations: 5
```

The two resulting models can be compared within a test which is making use of the fact the approximately, the sum of squared deviances is χ^2 distributed with $n-(p+1)$ degrees of freedom, where p is the number of explanatory variables.

```
> anova(mod.glm,modelglm,test="Chisq")

Analysis of Deviance Table

Model 1: diabetes ~ glucose + mass + age
Model 2: diabetes ~ pregnant + glucose + pressure + triceps + insulin +
    mass + pedigree + age
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       296     266.91
2       291     262.09  5   4.8196   0.4383
```

The null hypothesis is that the smaller Model 1 is the true one, and this cannot be rejected.

- *Prediction of the class membership and presentation of the results, see Figure 6.3*

```
> plot(predict(mod.glm, pid[-train,]),col= as.numeric(pid$diabetes[-train])+2)
```

Note that by default (of the function `predict.glm()`), the predictions are returned in the scale of the linear predictor, and thus zero is the decision boundary. One could also get predictions in the scale of the response variable (with `type="response"`).

- *Comparison of LDA and logistic regression*
- Logistic regression makes no assumption about the distribution. LDA assumes Gaussian distributions with equal covariances.
- A comparison of the LDA and logistic regression outcome is in Figure 6.4.

```
> modlda <- lda(diabetes~., data = pid[train,])
> plot(predict(mod.glm, pid[-train,]), col=as.numeric(pid$diabetes[-train])+2,
+      pch=as.numeric(predict(modlda,pid[-train,])$class))
```

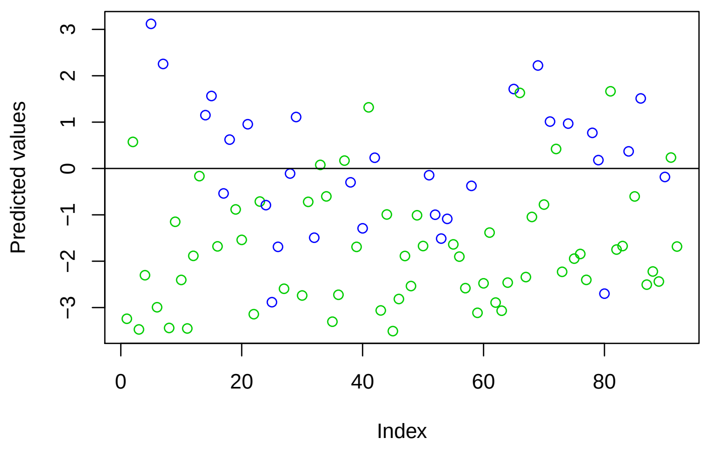

Figure 6.3: Prediction of the group memberships; the line is the separation from logistic regression, the color is the true group membership

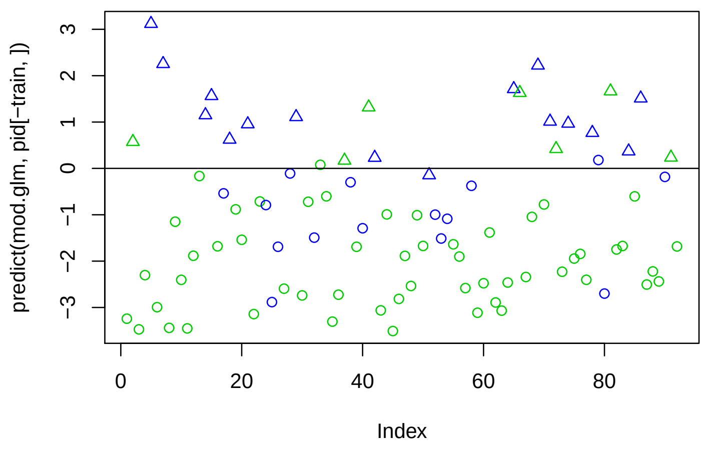

Figure 6.4: LDA (symbols) versus logistic regression (line); the color corresponds to the true classes

- *Prediction of the PimaIndianDiabetes data with logistic regression*

Here are again two evaluation schemes, the first based on cross-validation, the second based on bootstrap, and the final simple evaluation based on just one training and test set:

```
> library(ipred)
> mypred <- function(object, newdata) UseMethod("mypred", object)
> mypred.glm = function(object, newdata){
+     LEV = levels(object$model[,1])
+     as.factor(LEV[(predict(object, newdata=newdata, type="response")>0.5)+1])
+ }
> CEE <- control.errorest(k = 5, nboot=10)
> logcvest <- errorest(diabetes~., data=pid[train,], model=glm,
+                      family=binomial(),  predict=mypred, est.para=CEE)
> logcvest

Call:
errorest.data.frame(formula = diabetes ~ ., data = pid[train,
    ], model = glm, predict = mypred, est.para = CEE, family = binomial())

         5-fold cross-validation estimator of misclassification error

Misclassification error:  0.2133

> logbest <- errorest(diabetes~., data=pid[train,], model=glm,
+                     family=binomial(),  predict=mypred, estimator="boot", est.para=CEE)
> logbest

Call:
errorest.data.frame(formula = diabetes ~ ., data = pid[train,
    ], model = glm, predict = mypred, estimator = "boot", est.para = CEE,
    family = binomial())

         Bootstrap estimator of misclassification error
         with 10 bootstrap replications

Misclassification error:  0.2355
Standard deviation: 0.0103
```

Simpler (but less reliable) evaluation of the misclassification rate:

```
> TAB <- table(pid$diabetes[-train],mypred(mod.glm, pid[-train,]))
> mkrlog <- 1-sum(diag(TAB))/sum(TAB)
> mkrlog

[1] 0.25
```

	INDR	LDA	QDA	RDA	GLM
MKR	0.239	0.239	0.25	0.239	0.25