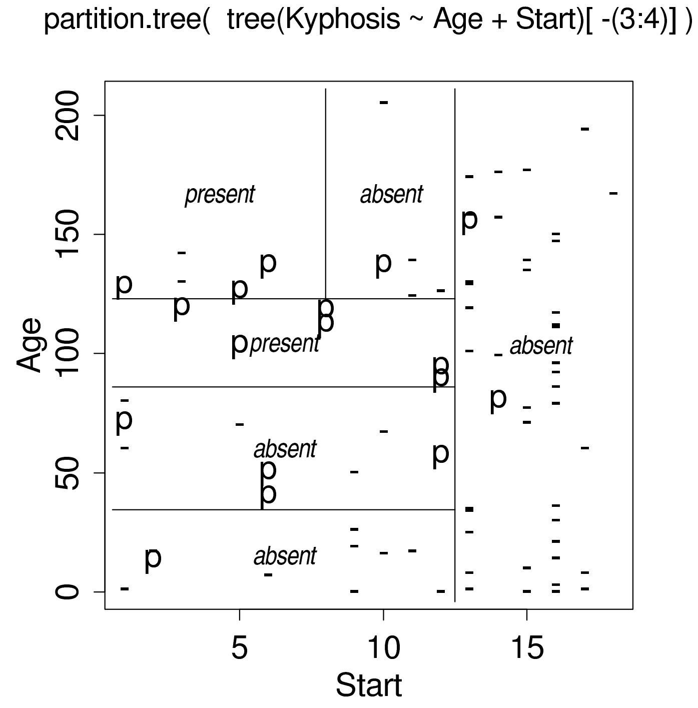

Figure 7.8: Partition of $\mathbb{R}^2$ for part of the kyphosis data set. "p" denotes presence of kyphosis, "-" denotes absence. Majority voting in each rectangle then determines the classification rule.

corresponds to a **recursive** partitioning scheme as shown in Figure 7.8.

The process of backward deletion in step 5 can now be more easily understood. Steps 1–4 result in a large tree $\mathcal{T}_M$ $(M = R - 1)$. Backward elimination, or tree pruning, then deletes successively the terminal node in the tree which increases the negative log-likelihood the least. This will produce a sequence of trees

$$\mathcal{T}_M \supset \mathcal{T}_{M-1} \supset \ldots \supset \mathcal{T}_1 = \{\mathcal{R}_0\}, \quad \mathcal{R}_0 = \text{root tree} = \mathbb{R}^p,$$

and we can select the "best" tree via crossvalidation as follows: As in regression, where we had a C_p criterion, the set of nested trees is here produced via so called "cost complexity pruning" (=: `cp`). The relevant measure is a penalized goodness of fit,

$$R_\alpha(\mathcal{T}) := R(\mathcal{T}) + \alpha \times \text{size}(\mathcal{T}), \tag{7.4}$$

where the *size* of a tree is the number of its leaves, or also 1+ the number of "cuts" and $R(.)$ is a quality of fit measure such as the deviance (sum of squares in the case of regression) or misclassification rate.

Now pruning is done such that for (conceptually) each α an optimally (wrt $R_\alpha(\cdot)$) pruned tree is chosen. For model selection, i.e., determining the amount of pruning, we now only need to choose the best α (or its normalization, `cp` = $\alpha/R(\mathcal{T}_\emptyset)$ in `rpart()`). For this, k-fold crossvalidation ($k = 10$ by default) is applied to compute CV error rates (in the case of classification) for each α. Instead of directly using the minimal error rate to choose α, the "1 S.E. rule" is used (see fig. 7.9): One takes the smallest tree (most sparse model) such that its error is at most one standard error larger than minimal one.

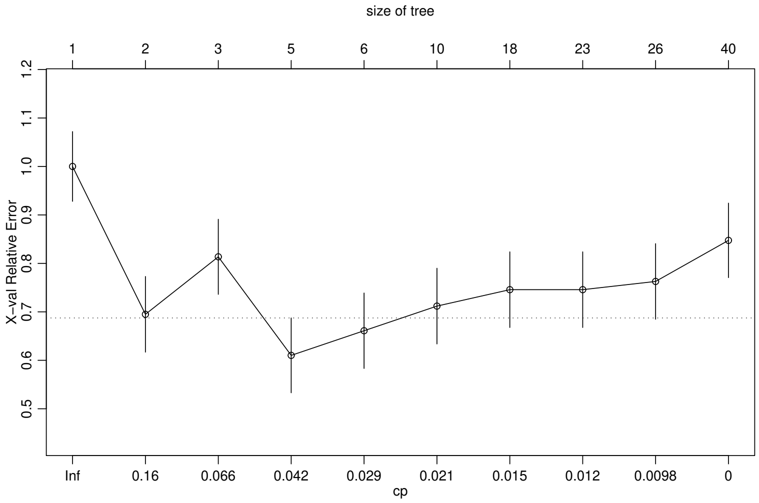

Figure 7.9: `plotcp(.)` for the Kyphosis example shows the full leave-1-out crossvalidated error rates of `rpart()`-models as function of α or `cp`, respectively. A tree with five leaves (final nodes) seems therefore optimal.

7.6.3 Pros and cons of trees

One of the very nice properties of trees is their interpretation. Displaying information in terms of a tree structure is extremely useful and very popular in very many scientific fields. From the view of trees it also becomes clear that trees with depth ℓ may include interaction terms of degree ℓ between different predictor variables. Regarding performance, classification trees often yield quite good prediction results.
Also note that there is a nice approach for dealing with *missing values* via so called "surrogate splits" which allow to predict also when some predictor variable values are not known. Finally, it may be worthwhile to point out that tree models/algorithms are doing variable selection automatically.

The disadvantages of trees include the following. The regression function estimate, or probability estimate in classification, is piecewise constant: this is not usually the form one thinks of an underlying "true" function. This also implies that the prediction accuracy for regression curve estimation (or probability estimation in classification) is often not among the best. Finally, the greedy tree-type algorithm produces fairly unstable splits: in particular, if one of the first splits is "wrong", everything below this split (in terms of the tree) will be "wrong". Thus, interpretation of a tree – despite its attractiveness – should be done with caution.

In recent years, trees have also been used as building blocks of much more flexible (and computationally expensive) fitting methodologies, such as boosting, see chapter 8, or in "random forests". There, many trees are fit to the data and their variable selection and predictions combined into a final model.

7.6.4 CART in R

The function `rpart` from the package `rpart` in R can be used for fitting classification or regression trees.
As an example, we run a regression tree for the ozone data set and display the tree in Figure 7.10.

```
> fit <- rpart(log(O3) ~ . , data= d.ozone, method = "anova")
```

```
                                        # method = ''anova'' specifies a regression tree
> fit
n= 330

node), split, n, deviance, yval
      * denotes terminal node

 1) root 330 184.251800 2.2129670
   2) temp< 65.5 198  67.348040 1.7804120
     4) ibht>=3669 100  25.313780 1.4871350
       8) temp< 57.5 78  16.626380 1.3713510
        16) humidity>=70 16   4.480741 0.9331287 *
        17) humidity< 70 62   8.280081 1.4844410 *
       9) temp>=57.5 22   3.934431 1.8976390 *
     5) ibht< 3669 98  24.656390 2.0796750
      10) day>=332 13   1.852759 1.3895210 *
      11) day< 332 85  15.664550 2.1852280
        22) day< 47.5 24   3.217857 1.7721080 *
        23) day>=47.5 61   6.739110 2.3477670 *
   3) temp>=65.5 132  24.287510 2.8617990
     6) vsty>=145 18   3.854144 2.2880300 *
     7) vsty< 145 114  13.571930 2.9523940
      14) ibtp< 226.5 57   4.525832 2.7703180 *
      15) ibtp>=226.5 57   5.266831 3.1344690 *

> library(maptree) ; draw.tree(fit)      ## is slightly better than simply
## plot(fit) ; text(fit, use.n = TRUE) ## for plotting the tree
```

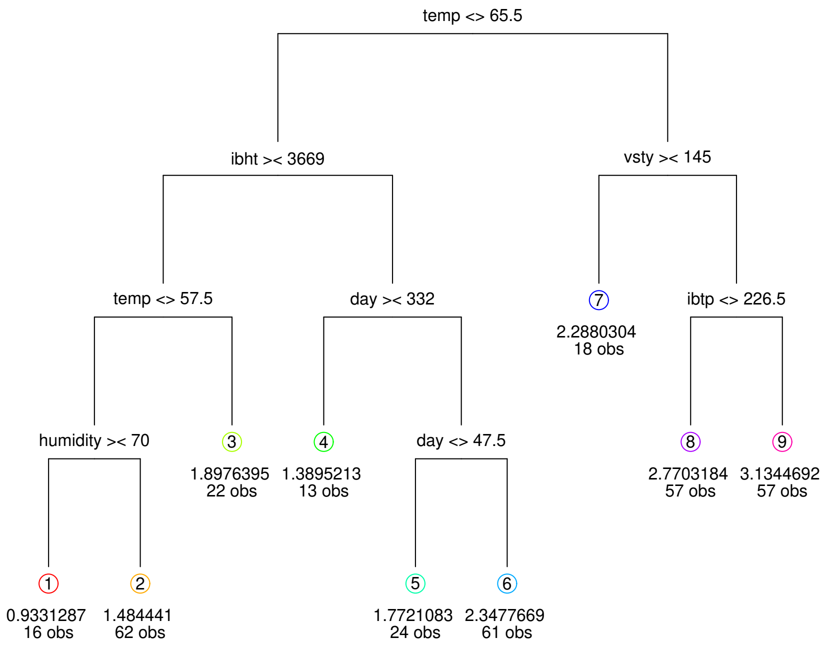

Figure 7.10: Regression tree for the ozone data set.

7.7 Variable Selection, Regularization, Ridging and the Lasso

7.7.1 Introduction

The following considerations and methods apply to both classification and regression in principle. For several reasons however we will focus on *regression* in this whole section.

In problems with many potential regressor or explanatory variables which may partly be highly correlated with each other, the classical least-squares regression approach can suffer from the fact that estimated regression coefficients can become quite ill-determined, i.e. have a high variance even when the fitted regression surface may be well determined. Also, in chapter 1's model equation (1.2) $Y_i = \mathbf{x}_i{}^\intercal \boldsymbol{\beta} + \varepsilon_i$, $(i = 1, \ldots, n)$, equivalently

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{7.5}$$

where $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, and $\mathbf{X}$ is the $n \times p$ matrix of rows $\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_n$ (or columns $\mathbf{x}^{(1)}$,...,$\mathbf{x}^{(p)}$), we had assumed that the number of observations n was larger than the number of parameters p, i.e., $n > p$, in order e.g, for the matrix $X^\intercal X$ to be of full rank p.

One main motivation this topic is the advent of data where there are many more potential explanatory variables than observations, in short, where $\boxed{p \gg n}$. Today the most important class of such applications is the analysis of genomic data, where e.g., for so called microarrays, one gets gene expression levels for thousands of genes (p = 1000–50'000) for typically a few dozen cells (n = 20–100). The target variable Y here is often binary (e.g., "normal" vs "special") which means a classification problem with $J = 2$ classes; we assume here however that treating it as a regression problem (e.g., using least squares or L_2 error) is useful.

7.7.2 Ridge Regression

Let's consider the regression model (7.5), or

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p} + \varepsilon_i, \quad i = 1, \ldots, n.$$

For least squares, one can see that $\hat{\beta}_0' = \overline{Y}(= 1/n \sum_{i=1}^n Y_i)$ when the model is re-written as

$$Y_i = \beta_0' + \beta_1(x_{i,1} - \overline{x_{.,1}}) + \ldots + \beta_p(x_{i,p} - \overline{x_{.,p}}) + \varepsilon_i.$$

Hence, in all of the following, we will assume that all the variables ($Y_.$ and $x_{.,j} =: \mathbf{x}^{(j)}$) have been centered, i.e., mean subtracted, such that no intercept is needed, and we use the equivalent model for the transformed variables,

$$Y_i = \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p} + \varepsilon_i, \quad i = 1, \ldots, n. \tag{7.6}$$

Note that in order have β_j on a comparable scale, one also typically *scales* the $\mathbf{x}^{(j)}$'s such that $\left\|\mathbf{x}^{(j)}\right\| = 1$ for all j .

Let's now assume that the variables (column vectors) $\mathbf{x}^{(j)} = (x_{i,j})_{i=1}^n$ and $\mathbf{x}^{(k)}$ are highly correlated (positively or negatively), or equivalently, since they both have mean 0, $\mathbf{x}^{(k)} \approx c\mathbf{x}^{(j)}$ (for $c \neq 0$) or $\mathbf{x}^{(j)} \approx 1/c\,\mathbf{x}^{(k)}$. As a consequence, in the following part of the model,

$$\begin{aligned} \beta_j \mathbf{x}^{(j)} + \beta_k \mathbf{x}^{(k)} &\approx \beta_j \mathbf{x}^{(j)} + \beta_k \left(c\mathbf{x}^{(j)}\right) \\ &= (\beta_j + \beta_k c)\mathbf{x}^{(j)} \\ &\approx (\beta_k + \beta_j / c)\mathbf{x}^{(k)} \end{aligned}$$