Computational Statistics

Summer 2004

Peter Bühlmann
Seminar für Statistik
ETH Zürich

April 1, 2004

Contents

Chapter 1

Multiple Linear Regression

1.1 Introduction

Linear regression is a widely used statistical model in a broad variety of applications. It is one of the easiest examples to demonstrate important aspects of statistical modelling.

1.2 The Linear Model

Multiple Regression Model:
Given is one **response variable**: up to some random errors
it is a linear function of several **predictors** (or **covariables**).
The linear function involves unknown parameters. The goal is to estimate these parameters, to study their relevance and to estimate the error variance.

Model formula:

$$Y_i = \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i \qquad (i = 1, \ldots, n) \tag{1.1}$$

Usually we assume that $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. with $\mathbb{E}[\varepsilon_i] = 0,\ \mathrm{Var}(\varepsilon_i) = \sigma^2$.

Notations:

- $\mathbf{Y} = \{Y_i; i = 1, \ldots, n\}$ is the vector of the **response variables**
- $\mathbf{x}^{(j)} = \{x_{ij}; i = 1, \ldots, n\}$ is the vector of the jth predictor (covariable) $(j = 1, \ldots, p)$
- $\mathbf{x}_i = \{x_{ij}; j = 1, \ldots p\}$ is the vector of predictors for the ith observation $(i = 1, \ldots, n)$
- $\beta = \{\beta_j; j = 1, \ldots, p\}$ is the vector of the unknown parameters
- $\varepsilon = \{\varepsilon_i; i = 1, \ldots, n\}$ is the vector of the unknown random **errors**
- n is the sample size, p is the number of predictors

The parameters β_j and σ^2 are unknown and the errors ε_i are unobservable. On the other hand, the response variables Y_i and the predictors x_{ij} have been observed.

Model in vector notation:

$$Y_i = \mathbf{x}_i^T \beta + \varepsilon_i \qquad (i = 1, \ldots, n)$$

Model in matrix form:

$$\begin{array}{ccccccc} \mathbf{Y} & = & X & \times & \beta & + & \varepsilon \\ n \times 1 & & n \times p & & p \times 1 & & n \times 1 \end{array}$$

where X is a $(n \times p)$-matrix with rows $\mathbf{x}_i^T$ and columns $\mathbf{x}^{(j)}$.

The first predictor variable is often a constant, i.e. $x_{i1} \equiv 1$ for all i. We then get an intercept in the model.

$$Y_i = \beta_1 + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i.$$

We typically assume that sample size n is larger than the number of predictors p ($p < n$) and moreover, that the matrix X has full rank p, i.e. the p column vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)}$ are linearly independent.

1.2.1 Stochastic Models

The linear model in (1.1) involves some stochastic (random) components: the error terms ε_i are random variables and hence the response variables Y_i as well. The predictor variables x_{ij} are here assumed to be non-random. But in some applications, it is more appropriate to treat the predictor variables as random variables.

The stochastic nature of the error terms ε_i can be assigned to various sources: for example, measurement errors or inability to capture all underlying non-systematic effects which are then summarized by a random variable with expectation zero. The stochastic modelling approach will allow to quantify uncertainty, to assign significance to various components, e.g. significance of predictor variables in model (1.1), and to find a good compromise between the size of a model and the ability to describe the data (see section 1.7).

The observed response in the data is always assumed to be realizations of the random variables $Y_1, \ldots, Y_n$; the x_{ij}'s are non-random and equal to the observed predictors in the data.

1.2.2 Examples

Two-sample model:

$$p = 2, \qquad X = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix}, \qquad \beta = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}.$$

Main questions: $\mu_1 = \mu_2$? Quantitative difference between μ_1 and μ_2?
From introductory courses we know that one could use the two-sample t-test or two-sample Wilcoxon test.

Regression through the origin: $Y_i = \beta x_i + \varepsilon_i$ $(i = 1, \ldots n)$.

$$p = 1, \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad \beta = \beta.$$

Simple linear regression: $Y_i = \beta_1 + \beta_2 x_i + \varepsilon_i$ $(i = 1, \ldots n)$.

$$p = 2 \qquad X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \qquad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}.$$

Quadratic regression: $Y_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2 + \varepsilon_i$ $(i = 1, \ldots n)$.

$$p = 3, \qquad X = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}, \qquad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}.$$

Note that the fitted function is quadratic in the x_i's but *linear* in the coefficients β_j and therefore a special case of the linear model (1.1).

Regression with transformed predictor variables:
$Y_i = \beta_1 + \beta_2 \log(x_{i2}) + \beta_3 \sin(\pi x_{i3}) + \varepsilon_i$ $(i = 1, \ldots n)$.

$$p = 3, \qquad X = \begin{pmatrix} 1 & \log(x_{12}) & \sin(\pi x_{13}) \\ 1 & \log(x_{22}) & \sin(\pi x_{23}) \\ \vdots & \vdots & \vdots \\ 1 & \log(x_{n2}) & \sin(\pi x_{n3}) \end{pmatrix}, \qquad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}.$$

Again, the model is *linear* in the coefficients β_j but nonlinear in the x_{ij}'s.

In summary:

> The model in (1.1) is called linear because it is linear in the coefficients β_j.
> The predictor (and also the response) variables can be transformed versions of the original predictor and/or response variables.

1.2.3 Goals of a linear regression analysis

- **A good "fit".** Fitting or estimating a (hyper-)plane over the predictor variables to explain the response variables such that the errors are "small". The standard tool for this is the method of *least squares* (see section 1.3.
- **Good parameter estimates.** This is useful to describe the change of the response when varying some predictor variable(s).
- **Good prediction.** This is useful to predict a new response as as a function of new predictor variables.

- **Uncertainties and significance for the three goals above.** Confidence intervals and statistical tests are useful tools for this goal.

- **Development of a good model.** In an interactive process, using methods for the goals mentioned above, we may change parts of an initial model to come up with a better model.

The first and third goal can become conflicting, see section 1.7.

1.3 Least Squares Method

We assume the linear model $\mathbf{Y} = X\beta + \varepsilon$. We are looking for a "good" estimate of β. The least squares estimator $\hat{\beta}$ is defined as

$$\hat{\beta} = \arg\min_{\beta} \|\mathbf{Y} - X\beta\|, \tag{1.2}$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^n$.

1.3.1 The normal equations

The minimizer in (1.2) can be computed explicitly (assuming that X has rank p). Computing partial derivatives of $\|\mathbf{Y} - X\beta\|^2$ to β (p-dimensional gradient vector), evaluated at $\hat{\beta}$, and setting them to zero yields the **normal equations**

$$(-2)\ X^T(\mathbf{Y} - X\hat{\beta}) = \mathbf{0} \qquad ((p \times 1) - \text{null-vector}).$$

Thus, we get

$$X^T X\hat{\beta} = X^T\mathbf{Y}.$$

These are the p linear normal equations for the p unknowns (components of $\hat{\beta}$).

Assuming that the matrix X has full rank p, the $p \times p$ matrix $X^T X$ is invertible. In this case, the least squares estimator is unique and can be represented as

$$\boxed{\hat{\beta} = (X^T X)^{-1} X^T \mathbf{Y}.}$$

This formula is useful for theoretical purposes. For numerical computation it is much more stable to use the QR decomposition instead of inverting the matrix $X^T X$.

The usual estimate for σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} r_i^2.$$

Note that the r_i's are estimates for ε_i's; hence the estimator is plausible, up to the somewhat unusual factor $1/(n-p)$. It will be shown in section 1.4.1 that due to this factor, $\mathbb{E}[\hat{\sigma}^2] = \sigma^2$.

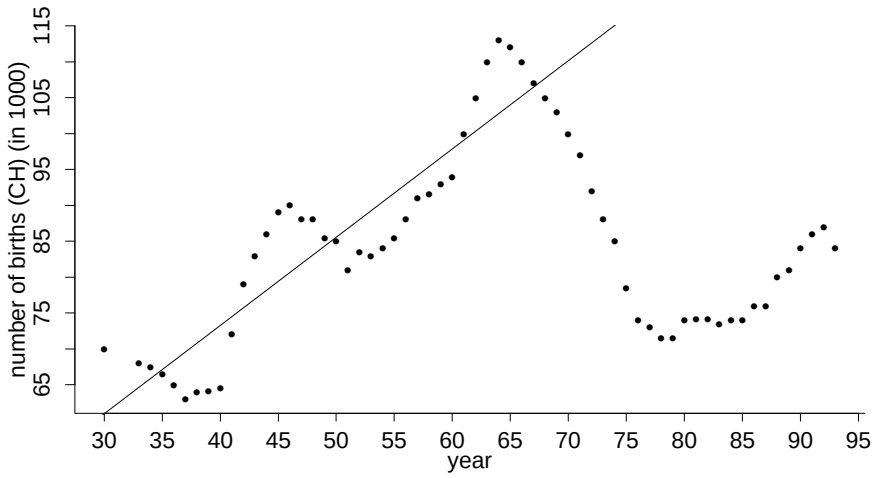


Figure 1.1: The pill kink.

1.3.2 Assumptions for the Linear Model

We emphasize here that we do not make any assumptions on the predictor variables, except that the matrix X has full rank $p < n$. In particular, the predictor variables can be continuous or discrete (e.g. binary).

We need some assumptions so that fitting a linear model by least squares is reasonable and that tests and confidence intervals are approximately valid.

1. **The linear regression equation is correct.** This means: $\mathbb{E}[\varepsilon_i] = 0$ for all i.
2. **All $\mathbf{x}_i$'s are exact.** This means that we can observe them perfectly.
3. **The variance of the errors is constant ("homoscedasticity").** This means: $\mathrm{Var}(\varepsilon_i) = \sigma^2$ for all i.
4. **The errors are uncorrelated.** This means: $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.
5. **The errors $\{\varepsilon_i; \imath = 1, \ldots, n\}$ are jointly normally distributed.** This implies that also $\{Y_i; i = 1, \ldots, n\}$ are jointly normally distributed.

In case of violations of item 3, we can use weighted least squares instead of least squares. Similarly, if item 4. is violated, we can use generalized least squares. If the normality assumption in 5. does not hold, we can use robust methods instead of least squares. If assumption 2. fails to be true, we need corrections known from "errors in variables" methods. If the crucial assumption in 1. fails, we need other models than the linear model.

The following example shows violations of assumption 1. and 4. The response variable is the annual number of birth in Switzerland since 1930, and the predictor variable is the time (year).

We see in Figure 1.1 that the data can be approximately described by a linear relation until the "pill kink" in 1964. We also see that the errors seem to be correlated: they are all positive or negative during periods of about 20 years. Finally, the linear model is not representative after the pill kink in 1964. In general, it is dangerous to use a fitted model for extrapolation where no predictor variables have been observed (for example: if

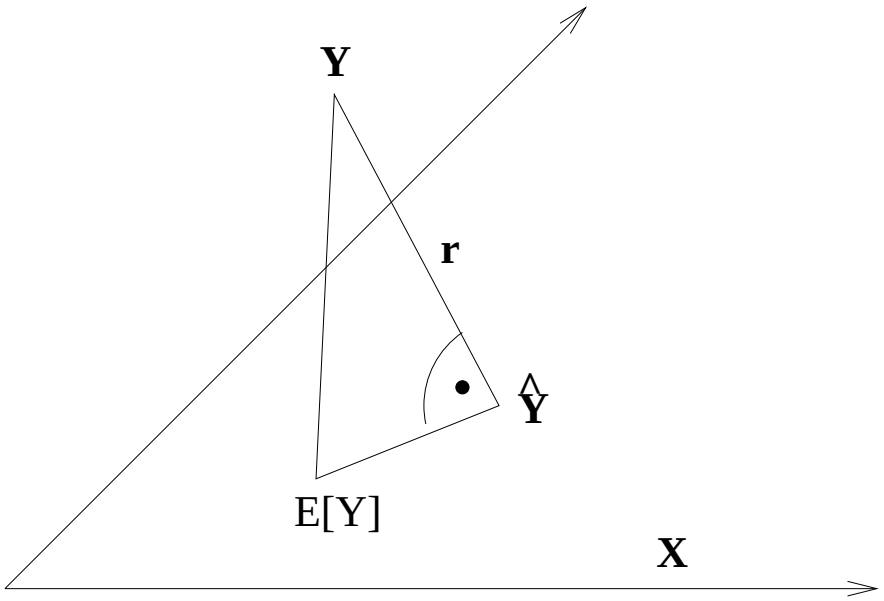


Figure 1.2: The residual vector $\mathbf{r}$ is orthogonal to $\mathcal{X}$.

we would have fitted the linear model in 1964 for prediction of number of births in the future until 2003).

1.3.3 Geometrical Interpretation

The response variable $\mathbf{Y}$ is a vector in $\mathbb{R}^n$. Also, $X\beta$ describes a p-dimensional subspace $\mathcal{X}$ in $\mathbb{R}^n$ (through the origin) when varying $\beta \in \mathbb{R}^p$ (assuming that X has full rank p). The least squares estimator $\hat{\beta}$ is then such that $X\hat{\beta}$ is closest to $\mathbf{Y}$ with respect to the Euclidean distance. But this means geometrically that

$$\boxed{X\hat{\beta} \text{ is the orthogonal projection of } \mathbf{Y} \text{ onto } \mathcal{X}.}$$

We denote the (vector of) fitted values by

$$\hat{\mathbf{Y}} = X\hat{\beta}.$$

They can be viewed as an estimate of $X\beta$.

The (vector of) residuals is defined by

$$\mathbf{r} = \mathbf{Y} - \hat{\mathbf{Y}}.$$

Geometrically, it is evident that the residuals are orthogonal to $\mathcal{X}$, because $\hat{\mathbf{Y}}$ is the orthogonal projection of $\mathbf{Y}$ onto $\mathcal{X}$. This means that

$$\mathbf{r}^T\mathbf{x}^{(j)} = 0 \text{ for all } j = 1, \ldots, p,$$

where $\mathbf{x}^{(j)}$ is the jth column of X.

We can formally see why the map

$$\mathbf{Y} \mapsto \hat{\mathbf{Y}}$$

is an orthogonal projection. Since $\hat{\mathbf{Y}} = X\hat{\beta} = X(X^TX)^{-1}X^T\mathbf{Y}$, the map can be represented by the matrix

$$P = X(X^TX)^{-1}X^T.$$

It is evident that P is symmetric ($P^T = P$) and P is idem-potent ($P^2 = P$). Furthermore

$$\sum_i P_{ii} = tr(P) = tr(X(X^TX)^{-1}X^T) = tr((X^TX)^{-1}X^TX) = tr(I_{p\times p}) = p.$$

But these 3 properties characterize that P is an orthogonal projection from $\mathbb{R}^n$ onto a p-dimensional subspace, here $\mathcal{X}$.

The residuals $\mathbf{r}$ can be represented as

$$\mathbf{r} = (I - P)Y,$$

where $I - P$ is now also an orthogonal projection onto the orthogonal complement of $\mathcal{X}$, $\mathcal{X}^\perp = \mathbb{R}^n \setminus \mathcal{X}$, which is $(n-p)$-dimensional. In particular, the residuals are elements of $\mathcal{X}^\perp$.

1.3.4 Don't do many regressions on single variables!

In general, it is not appropriate to replace multiple regression by many single regressions (on single predictor variables). The following (synthetic) example should help to demonstrate this point.

Consider two predictor variables $x^{(1)}, x^{(2)}$ and a response variable Y with the values

$x^{(1)}$	0	1	2	3	0	1	2	3
$x^{(2)}$	−1	0	1	2	1	2	3	4
Y	1	2	3	4	−1	0	1	2

Multiple regression yields the least squares solution which describes the data points exactly

$$Y_i = \hat{Y}_i = 2x_{i1} - x_{i2} \text{ for all } i \qquad (\hat{\sigma}^2 = 0). \tag{1.3}$$

The coefficients 2 and -1, respectively, describe how y is changing when varying either $x^{(1)}$ or $x^{(2)}$ and keeping the other predictor variable constant. In particular, we see that Y decreases when $x^{(2)}$ increases.

On the other hand, if we do a simple regression of Y onto $x^{(2)}$ (while ignoring the values of $x^{(1)}$; and thus, we do not keep them constant), we obtain the least squares estimate

$$\hat{Y}_i = \frac{1}{9}x_{i2} + \frac{4}{3} \text{ for all i} \qquad (\hat{\sigma}^2 = 1.72).$$

This least squares regression line describes how Y changes when varying $x^{(2)}$ while ignoring $x^{(1)}$. In particular, $\hat{Y}$ increases when $x^{(2)}2$ increases, in contrast to multiple regression!

The reason for this phenomenon is that $x^{(1)}$ and $x^{(2)}$ are strongly correlated: if $x^{(2)}$ increases, then also $x^{(1)}$ increases. Note that in the multiple regression solution, $x^{(1)}$ has a larger coefficient in absolute value than $x^{(2)}$ and hence, an increase in $x^{(1)}$ has a stronger influence for changing y than $x^{(2)}$. The correlation among the predictors in general makes also the interpretation of the regression coefficients more subtle: in the current setting, the coefficient β_1 quantifies the influence of $x^{(1)}$ on Y *after* having subtracted the effect of $x^{(2)}$ on Y, see also section 1.5.

Summarizing:

> Simple least squares regressions on single predictor variables yield the multiple regression least squares solution, if the predictor variables are orthogonal.
> In general, multiple regression is the appropriate tool
> to include effects of more than one predictor variables simultaneously.

The equivalence in case of orthogonal predictors is easy to see algebraically. Orthogonality of predictors means $X^T X = \text{diag}(\sum_{i=1}^n x_{i1}^2, \ldots, \sum_{i=1}^n x_{ip}^2)$ and hence the least squares estimator

$$\hat{\beta}_j = \sum_{i=1}^n x_{ij} Y_i / \sum_{i=1}^n x_{ij}^2 \ (j = 1, \ldots, p),$$

i.e. $\hat{\beta}_j$ depends only on the the response variable Y_i and the jth predictor variable x_{ij}.

1.3.5 Computer-Output from R: Part I

We show here parts of the computer output (from R) when fitting a linear model to data about quality of asphalt.

```
   y = LOGRUT  : log("rate of rutting") = log(change of rut depth in inches
                    per million wheel passes)
               ["rut":= 'Wagenspur", ausgefahrenes Geleise]
  x1 = LOGVISC : log(viscosity of asphalt)
  x2 = ASPH : percentage of asphalt in surface course
  x3 = BASE : percentage of asphalt in  base   course
  x4 = RUN  : '0/1' indicator for two sets of runs.
  x5 = FINES: 10* percentage of fines in surface course
  x6 = VOIDS: percentage of voids in surface course
```

The following table shows the least squares estimates $\hat{\beta}_j$ $(j = 1, \ldots, 6)$, some empirical quantiles of the residuals r_i $(i = 1, \ldots, n)$, the estimated standard error of the errors (residuals) $\sqrt{\hat{\sigma}^2}$ and the so-called degrees of freedom $n - p$.

```
Call:
lm(formula = LOGRUT ~ ., data = asphalt1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.48348 -0.14374 -0.01198  0.15523  0.39652

Coefficients:
             Estimate
(Intercept) -5.781239
LOGVISC       -0.513325
ASPH          1.146898
BASE          0.232809
RUN          -0.618893
FINES         0.004343
VOIDS         0.316648

Residual standard error: 0.2604 on 24 degrees of freedom
```

1.4 Properties of Least Squares Estimates

As an introductory remark, we point out that the least squares estimates are random variables: for new data from the same data-generating mechanism, the data would look differently every time and hence also the least squares estimates. Figure 1.3 displays three least squares regression lines which are based on three different realizations from the same data-generating model (i.e. three simulations from a model). We see that the estimates are varying: they are random themselves!

1.4.1 Moments of least squares estimates

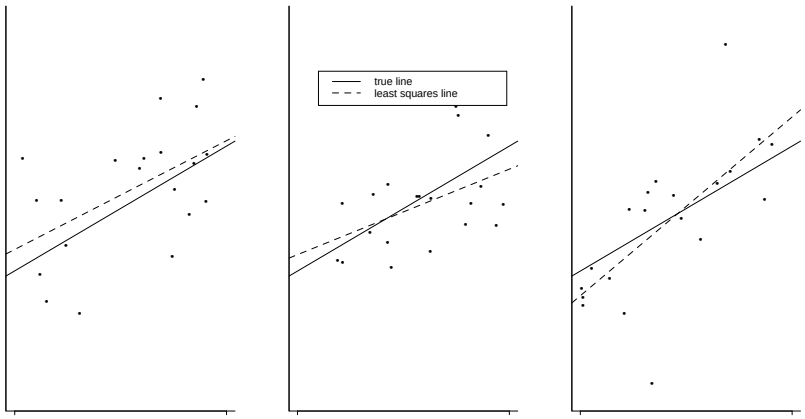


Figure 1.3: Three least squares estimated regression lines for three different data realizations from the same model.

We assume here the usual linear model

$$\mathbf{Y} = X\beta + \varepsilon,\ \mathbb{E}[\varepsilon] = 0,\ \mathrm{Cov}(\varepsilon) = \mathbb{E}[\varepsilon\varepsilon^T] = \sigma^2 I_{n\times n}. \tag{1.4}$$

This means that assumption 1.-4. from section 1.3.2 are satisfied.

It can then be shown that:

(i) $\mathbb{E}[\hat{\beta}] = \beta$: that is, $\hat{\beta}$ is **unbiased**

(ii) $\mathbb{E}[\hat{\mathbf{Y}}] = \mathbb{E}[\mathbf{Y}] = X\beta$ which follows from (i). Moreover, $\mathbb{E}[\mathbf{r}] = 0$.

(iii) $\mathrm{Cov}(\hat{\beta}) = \sigma^2(X^TX)^{-1}$

(iv) $\mathrm{Cov}(\hat{\mathbf{Y}}) = \sigma^2 P$, $\mathrm{Cov}(\mathbf{r}) = \sigma^2(I - P)$

The residuals (which are estimates of the unknown errors ε_i) are also having expectation zero but they are not uncorrelated:

$$\mathrm{Var}(r_i) = \sigma^2(1 - P_{ii}).$$

From this, we obtain

$$\mathbb{E}[\sum_{i=1}^n r_i^2] = \sum_{i=1}^n \mathbb{E}[r_i^2] = \sum_{i=1}^n \mathrm{Var}(r_i) = \sigma^2 \sum_{i=1}^n (1 - P_{ii}) = \sigma^2(n - \mathrm{tr}(P)) = \sigma^2(n-p).$$

Therefore, $\mathbb{E}[\hat{\sigma}^2] = \mathbb{E}[\sum_{i=1}^n r_i^2/(n-p)] = \sigma^2$ is **unbiased**.

1.4.2 Distribution of least squares estimates assuming Gaussian errors

We assume the linear model as in (1.4) but require in addition that $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d. $\sim \mathcal{N}(0, \sigma^2)$. It can then be shown that:

(i) $\hat{\beta} \sim \mathcal{N}_p(\beta, \sigma^2 (X^T X)^{-1})$

(ii) $\hat{\mathbf{Y}} \sim \mathcal{N}_n(X\beta, \sigma^2 P)$, $\mathbf{r} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 (I - P))$

(iii) $\hat{\sigma}^2 \sim \frac{\sigma^2}{n-p} \chi^2_{n-p}$.

The normality assumptions of the errors ε_i is often not (approximately) fulfilled in practice. We can then rely on the central limit theorem which implies that for large sample size n, the properties (i)-(iii) above are still approximately true. This is the usual justification in practice to use these properties for constructing confidence intervals and tests for the linear model parameters. However, it is often much better to use **robust methods** in case of non-Gaussian errors which we are not discussing here.

1.5 Tests and Confidence Regions

We assume the linear model as in (1.4) with $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d. $\sim \mathcal{N}(0, \sigma^2)$ (or with ε_i's i.i.d. and "large" sample size n). As we have seen above, the parameter estimates $\hat{\beta}$ are normally distributed.

If we are interested whether the jth predictor variable is relevant, we can test the null-hypothesis $H_{0,j} : \beta_j = 0$ against the alternative $H_{A,j} : \beta_j \neq 0$. We can then easily derive from the normal distribution of $\hat{\beta}_j$ that

$$\frac{\hat{\beta}_j}{\sqrt{\sigma^2 (X^T X)^{-1}_{jj}}} \sim \mathcal{N}(0,1) \text{ under the null-hypothesis } H_{0,j}.$$

Since σ^2 is unknown, this quantity is not useful, but if we substitute it with the estimate $\hat{\sigma}^2$ we obtain the so-called t-test statistic

$$T_j = \frac{\hat{\beta}_j}{\sqrt{\hat{\sigma}^2 (X^T X)^{-1}_{jj}}} \sim t_{n-p} \text{ under the hull-hypothesis } H_{0,j}, \tag{1.5}$$

which has a slightly different distribution than standard Normal $\mathcal{N}(0,1)$. The corresponding test is then called the t-test. In practice, we can thus quantify the relevance of individual predictor variables by looking at the size of the test-statistics T_j $(j = 1, \ldots, p)$ or at the corresponding P-values which may be more informative.

The problem by looking at *individual* tests $H_{0,j}$ is (besides the multiple testing problem in general) that it can happen that all individual tests do not reject the null-hypotheses (say at the 5% significance level) although it is true that some predictor variables have a significant effect. This "paradox" can occur because of correlation among the predictor variables.

> An individual t-test for $H_{0,j}$ should be interpreted as quantifying the effect of the jth predictor variable after having subtracted the linear effect of all other predictor variables on Y.

To test whether there exists *any* effect from the predictor variables, we can look at the simultaneous hull-hypothesis $H_0 : \beta_2 = \ldots = \beta_p = 0$ versus the alternative $H_A : \beta_j \neq$

0 for at least one $j \in \{2, \ldots, p\}$; we assume here that the first predictor variable is the constant $X_{i,1} \equiv 1$ (there are $p-1$ (non-trivial) predictor variables). Such a test can be developed with an analysis of variance decomposition which takes a simple form for this special case:

$$\|\mathbf{Y} - \overline{\mathbf{Y}}\|^2 = \|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2 + \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2$$

which decomposes the total squared error $\mathbf{Y} - \overline{\mathbf{Y}}$ around the mean $\overline{\mathbf{Y}} = n^{-1} \sum_{i=1}^n Y_i \cdot \mathbf{1}$ as a sum of the squared error due to the regression $\hat{\mathbf{Y}} - \overline{\mathbf{Y}}$ (the amount that the fitted values vary around the global arithmetic mean) and the squared residual error $\mathbf{r} = \mathbf{Y} - \hat{\mathbf{Y}}$. (The equality can be seen most easily from a geometrical point of view: the residuals $\mathbf{r}$ are orthogonal to $\mathcal{X}$ and hence to $\hat{\mathbf{Y}} - \overline{\mathbf{Y}}$). Such a decomposition is usually summarized by an ANOVA table (**AN**alysis **O**f **VA**riance).

	sum of squares	degrees of freedom	mean square	$\mathbf{E}$ [mean square]
regression	$\|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2$	$p-1$	$\|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2/(p-1)$	$\sigma^2 + \frac{\|\mathbf{E}[\mathbf{Y}] - \mathbf{E}[\overline{\mathbf{Y}}]\|^2}{p-1}$
error	$\|\mathbf{Y} - \hat{\mathbf{Y}}\|^2$	$n-p$	$\|\mathbf{Y} - \hat{\mathbf{Y}}\|^2/(n-p)$	σ^2
total around global mean	$\|\mathbf{Y} - \overline{\mathbf{Y}}\|^2$	$n-1$	–	–

In case of the global null-hypothesis, there is no effect of any predictor variable and hence $\mathbb{E}[\mathbf{Y}] \equiv const. = \mathbb{E}[\overline{\mathbf{Y}}]$: therefore, the expected mean square equals σ^2 under H_0. The idea is now to divide the mean square by the estimate $\hat{\sigma}^2$ to obtain a scale-free quantity: this leads to the so-called F-statistic

$$F = \frac{\|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2/(p-1)}{\|\mathbf{Y} - \hat{\mathbf{Y}}\|^2/(n-p)} \sim F_{p-1,n-p} \text{ under the global null-hypothesis } H_0.$$

This test is called the F-test (it is one among several other F-tests in regression).

Besides performing a global F-test to quantify the statistical significance of the predictor variables, we often want to describe the goodness of fit of the linear model for explaining the data. A meaningful quantity is the coefficient of determination, abbreviated by R^2,

$$R^2 = \frac{\|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2}{\|\mathbf{Y} - \overline{\mathbf{Y}}\|^2}$$

which is the proportion of the total variation of $\mathbf{Y}$ around $\overline{\mathbf{Y}}$ which is explained by the regression (see the ANOVA decomposition and table above).

Similarly to the t-tests as in (1.5), one can derive confidence intervals for the unknown parameters β_j:

$$\hat{\beta}_j \pm \sqrt{\hat{\sigma^2}(X^T X)^{-1}_{jj}}\, t_{n-p;1-\alpha/2}$$

is a two-sided confidence interval which covers the true β_j with probability $1-\alpha$; here, $t_{n-p;1-\alpha/2}$ denotes the $1-\alpha/2$ quantile of a t_{n-p} distribution.

1.5.1 Computer-Output from R: Part II

We consider again the dataset from section 1.3.5. We now give the complete list of summary statistics from a linear model fit to the data.

```
Call:
lm(formula = LOGRUT ~ ., data = asphalt1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.48348 -0.14374 -0.01198  0.15523  0.39652

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.781239   2.459179  -2.351 0.027280 *
LOGVISC     -0.513325   0.073056  -7.027 2.90e-07 ***
ASPH         1.146898   0.265572   4.319 0.000235 ***
BASE         0.232809   0.326528   0.713 0.482731
RUN         -0.618893   0.294384  -2.102 0.046199 *
FINES        0.004343   0.007881   0.551 0.586700
VOIDS        0.316648   0.110329   2.870 0.008433 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2604 on 24 degrees of freedom
Multiple R-Squared: 0.9722,     Adjusted R-squared: 0.9653
F-statistic: 140.1 on 6 and 24 DF,  p-value: < 2.2e-16
```

The table displays the standard errors of the estimates $\sqrt{\widehat{\mathrm{Var}}(\hat{\beta}_j)} = \sqrt{\hat{\sigma}^2 (X^T X)^{-1}_{jj}}$, the t-test statistics for the null-hypotheses $H_{0,j}: \beta_j = 0$ and their corresponding two-sided P-values with some abbreviation about strength of significance. Moreover, the R^2 and adjusted R^2 are given and finally also the F-test statistic for the null-hypothesis $H_0: \beta_2 = \ldots = \beta_p = 0$ (with the degrees of freedom) and its corresponding P-value.

1.6 Analysis of residuals and checking of model assumptions

The residuals $r_i = Y_i - \hat{Y}_i$ can serve as an approximation of the unobservable error term ε_i and for checking whether the linear model is appropriate.

1.6.1 The Tukey-Anscombe Plot

The Tukey-Anscombe is a graphical tool: we plot the residuals r_i (on the y-axis) versus the fitted values $\hat{Y}_i$ (on the x-axis). A reason to plot against the fitted values $\hat{Y}_i$ is that the sample correlation between r_i and $\hat{Y}_i$ is always zero.

In the ideal case, the points in the Tukey-Anscombe plot "fluctuate randomly" around the horizontal line through zero: see also Figure 1.4. An often encountered deviation is non-constant variability of the residuals, i.e. an indication that the variance of ε_i increases with the response variable Y_i: this is shown in Figure 1.5 a)–c). If the Tukey-Anscombe plot shows a trend, there is some evidence that the linear model assumption is not correct (the expectation of the error is not zero which indicates a systematic error): Figure 1.5d) is a typical example.

In case where the Tukey-Anscombe plot exhibits a systematic relation of the variability on the fitted values $\hat{Y}_i$, we should either transform the response variable or perform a weighted regression (see Section 1.6.4). If the standard deviation grows linearly with the

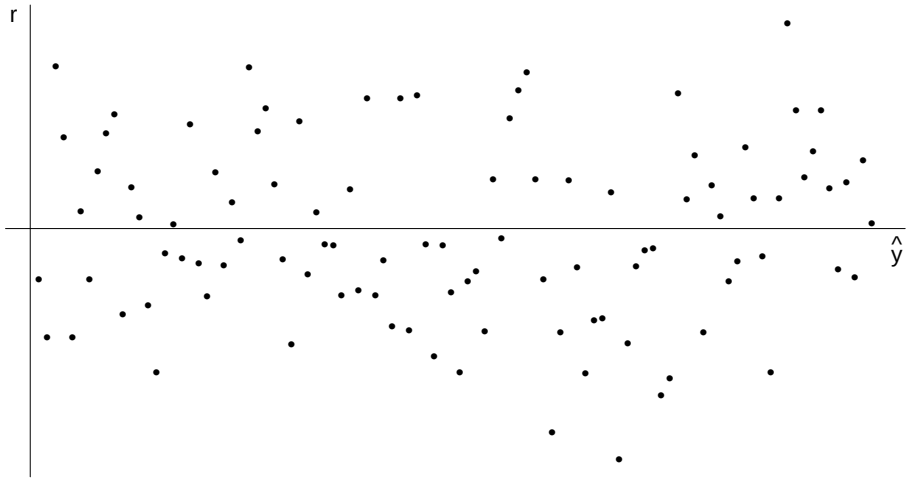


Figure 1.4: Ideal Tukey-Anscombe plot: no violations of model assumptions.

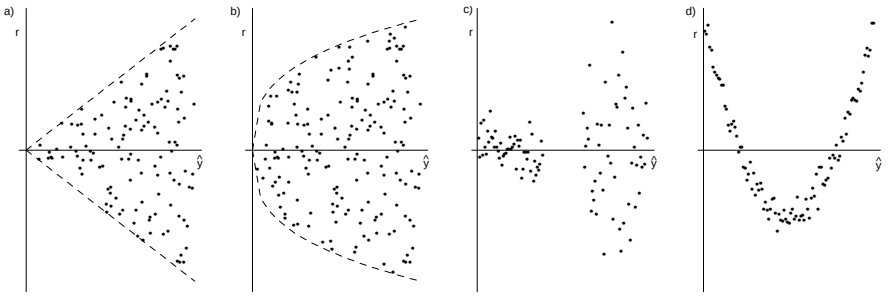


Figure 1.5: a) linear increase of standard deviation, b) nonlinear increase of standard deviation, c) 2 groups with different variances, d) missing quadratic term in the model.

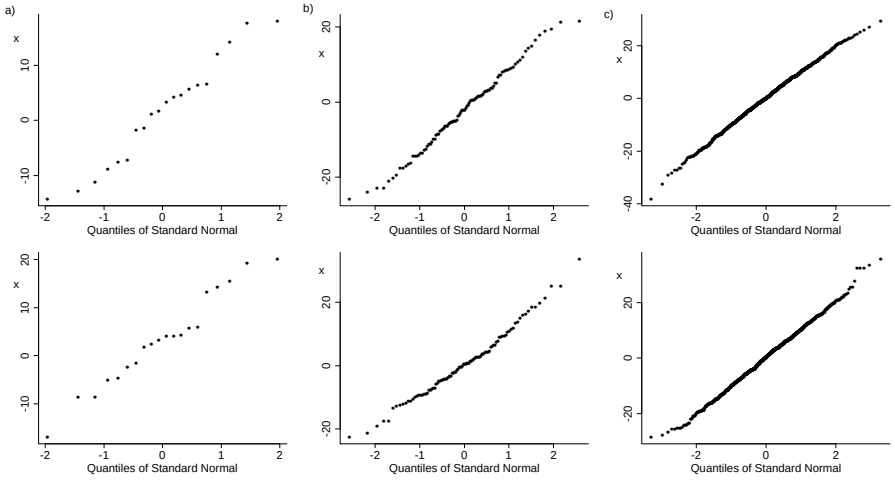


Figure 1.6: QQ-plots for i.i.d. normally distributed random variables. Two plots for each sample size n equal to a) 20, b) 100 and c) 1000.

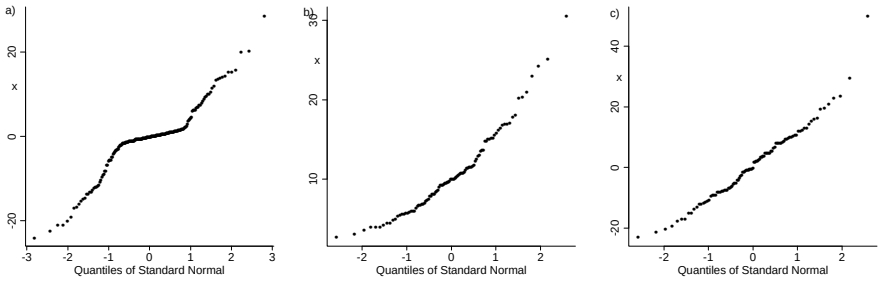


Figure 1.7: QQ-plots for a) long-tailed distribution, b) skewed distribution, c) dataset with outlier.

fitted values (as in Figure 1.5a)), the log-transform $Y \mapsto \log(Y)$ stabilizes the variance; if the standard deviation grows as the square root with the values $\hat{Y}_i$ (as in Figure 1.5a)), the square root transformation $Y \mapsto \sqrt{Y}$ stabilizes the variance.

1.6.2 The Normal Plot

Assumptions for the distribution of random variables can be graphically checked with the QQ (quantile-quantile) plot. In the special case of checking for the normal distribution, the QQ plot is also referred to as a normal plot.

In the linear model application, we plot the empirical quantiles of the residuals (on the y axis) versus the theoretical quantiles of a $\mathcal{N}(0,1)$ distribution (on the x axis). If the residuals would be normally distributed with expectation μ and variance σ^2, the normal plot would exhibit an approximate straight line with intercept μ and slope σ. Figures 1.6 and 1.7 show some normal plots with exactly normally and non-normally distributed observations.

1.6.3 Plot for detecting serial correlation

For checking independence of the errors we plot the residuals r_i versus the observation number i (or if available, the time t_i of recording the ith observation). If the residuals

vary randomly around the zero line, there are no indications for serial correlations among the errors ε_i. On the other hand, if neighboring (with respect to the x-axis) residuals look similar, the independence assumption for the errors seems violated.

1.6.4 Generalized least squares and weighted regression

This is currently under construction.

1.7 Model Selection

We assume the linear model

$$Y_i = \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i \ (i = 1, \ldots, n),$$

where $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d., $\mathbb{E}[\varepsilon_i] = 0$, $\mathrm{Var}(\varepsilon_i) = \sigma^2$.

Problem: Which of the predictor variables should be used in the linear model?
It may be that not all of the p predictor variables are relevant. In addition, every coefficient has to be estimated and thus is afflicted with variability: the individual variabilities for each coefficient sum up and **the variability of the estimated hyper-plane increases the more predictors are entered into the model, whether they are relevant or not**. The aim is often to look for the **optimal** or **best** - not the true - model.

What we just explained in words can be formalized a bit more. Suppose we are looking for optimizing the prediction

$$\sum_{r=1}^{q} \hat{\beta}_{j_r} x_{ij_r}$$

which includes q predictor variables with indices $j_1, \ldots, j_q \in \{1, \ldots, p\}$. The average mean squared error of this prediction is

$$\begin{aligned} & n^{-1} \sum_{i=1}^{n} \mathbb{E}[(m(\mathbf{x}_i) - \sum_{r=1}^{q} \hat{\beta}_{j_r} x_{ij_r})^2] \\ = \ & n^{-1} \sum_{i=1}^{n} (\mathbb{E}[\sum_{r=1}^{q} \hat{\beta}_{j_r} x_{ij_r}] - m(\mathbf{x}_i))^2 + \underbrace{n^{-1} \sum_{i=1}^{n} \mathrm{Var}(\sum_{r=1}^{q} \hat{\beta}_{j_r} x_{ij_r})}_{= \frac{q}{n}\sigma^2}, \end{aligned} \tag{1.6}$$

where $m(\cdot)$ denotes the regression function in the full model with all the predictor variables. It is plausible that the systematic error (squared bias) $n^{-1} \sum_{i=1}^{n} (\mathbb{E}[\sum_{r=1}^{q} \hat{\beta}_{j_r} x_{ij_r}] - m(\mathbf{x}_i))^2$ decreases as the number of predictor variables q increases (i.e. with respect to bias, we have nothing to lose by using as many predictors as we can), but the variance term increases linearly in the number of predictors q (the variance term equals $q/n\sigma^2$ which is not too difficult to derive). This is the so-called **bias-variance trade-off** which is present in very many other situations and applications in statistics. Finding the best model thus means to optimize the bias-variance trade-off: this is sometimes also referred to as "regularization" (for avoiding a too complex model).

1.7.1 Mallows C_p statistic

The mean squared error in (1.6) is unknown: we do not know the magnitude of the bias term but fortunately, we can estimate the mean squared error.

Denote by $SSE(\mathcal{M})$ the residual sum of squares in a model $\mathcal{M}$: it is overly optimistic and not a good measure to estimate the mean squared error in (1.6). For example, $SSE(\mathcal{M})$ becomes smaller the bigger the model $\mathcal{M}$ and the biggest model under consideration has the lowest SSE (which generally contradicts the equation in (1.6)).

For any (sub-)model $\mathcal{M}$ which involves some (or all) of the predictor variables, the mean squared error in (1.6) can be estimated by

$$n^{-1}SSE(\mathcal{M}) - \hat{\sigma}^2 + 2\hat{\sigma}^2|\mathcal{M}|/n,$$

where $\hat{\sigma}^2$ is the error variance estimate in the full model and $SSE(\mathcal{M})$ is the residual sum of squares in the submodel $\mathcal{M}$. (A justification can be found in the literature). Thus, in order to estimate the best model, we could search for the sub-model $\mathcal{M}$ minimizing the above quantity. Since $\hat{\sigma}^2$ and n are constants with respect to submodels $\mathcal{M}$, we can also consider the well-known C_p statistic

$$C_p(\mathcal{M}) = \frac{SSE(\mathcal{M})}{\hat{\sigma}^2} - n + 2|\mathcal{M}|$$

and search for the sub-model $\mathcal{M}$ minimizing the C_p statistic.

Other popular criteria to estimate the predictive potential of an estimated model are Akaike's information criterion (AIC) and the Bayesian information criterion (BIC), see section - "under construction" - .

Searching for the best model with respect to C_p

If the full model has p predictor variables, there are $2^p - 1$ sub-models (every predictor can be in or out but we exclude the sub-model $\mathcal{M}$ which corresponds to the empty set).

Therefore, an exhaustive search for the sub-model $\mathcal{M}$ minimizing the C_p statistic is only feasible if p is less than say 16 ($2^{16} - 1 = 65'535$ which is already fairly large). If p is "large", we can proceed with stepwise algorithms.

Forward selection.
1. Start with the smallest model $\mathcal{M}_0$ (location model) as the current model.
2. Include the predictor variable to the current model which reduces the residual sum of squares most.
3. Continue step 2. until all predictor variables have been chosen or until a large number of predictor variables has been selected. This produces a sequence of sub-models $\mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \ldots$
4. Choose the model in the sequence $\mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \ldots$ which has smallest C_p statistic.

Backward selection.
1. Start with the full model $\mathcal{M}_0$ as the current model.
2. Exclude the predictor variable from the current model which increases the residual sum of squares the least.
3. Continue step 2. until all predictor variables have been deleted (or a large number of predictor variables). This produces a sequence of sub-models $\mathcal{M}_0 \supseteq \mathcal{M}_1 \supseteq \mathcal{M}_2 \supseteq \ldots$
4. Choose the model in the sequence $\mathcal{M}_0 \supseteq \mathcal{M}_1 \supseteq \mathcal{M}_2 \supseteq \ldots$ which has smallest C_p statistic.

Backward selection is typically a bit better than forward selection but it is computationally more expensive. Also, in case where $p \geq n$, we don't want to fit the full model and forward selection is an appropriate way to proceed.

Chapter 2

Nonparametric Density Estimation

2.1 Introduction

For a moment, we will go back to simple data structures: we have observations which are realizations of univariate random variables,

$$X_1, \ldots, X_n \text{ i.i.d. } \sim F,$$

where F denotes an unknown cumulative distribution function. The goal is to estimate the distribution F. In particular, we are interested in estimating the density $f = F'$, assuming that it exists.

Instead of assuming a parametric model for the distribution (e.g. Normal distribution with unknown expectation and variance), we rather want to be "as general as possible": that is, we only assume that the density exists and is suitably smooth (e.g. differentiable). It is then possible to estimate the unknown density **function** $f(\cdot)$. Mathematically, a function is an **infinite-dimensional** object. Density estimation will become a "basic principle" how to do estimation for infinite-dimensional objects. We will make use of such a principle in many other settings such as nonparametric regression with one predictor variable (Chapter 3) and flexible regression and classification methods with many predictor variables (Chapter - "under construction" -).

2.2 Estimation of a density

We consider the data which records the duration of eruptions of "Old Faithful", a famous geysir in Yellowstone National Park (Wyoming, USA). You can watch via web-cam on `http://www.nps.gov/yell/oldfaithfulcam.htm`

2.2.1 Histogram

The histogram is the oldest and most popular density estimator. We need to specify an *"origin"* x_0 and the *class width* h for the specifications of the intervals

$$I_j = (x_0 + j \cdot h, x_0 + (j+1)h] \ \ (j = \ldots, -1, 0, 1, \ldots)$$

for which the histogram counts the number of observations falling into each I_j: we then plot the histogram such that the area of each bar is proportional to the number of observations falling into the corresponding class (interval I_j).

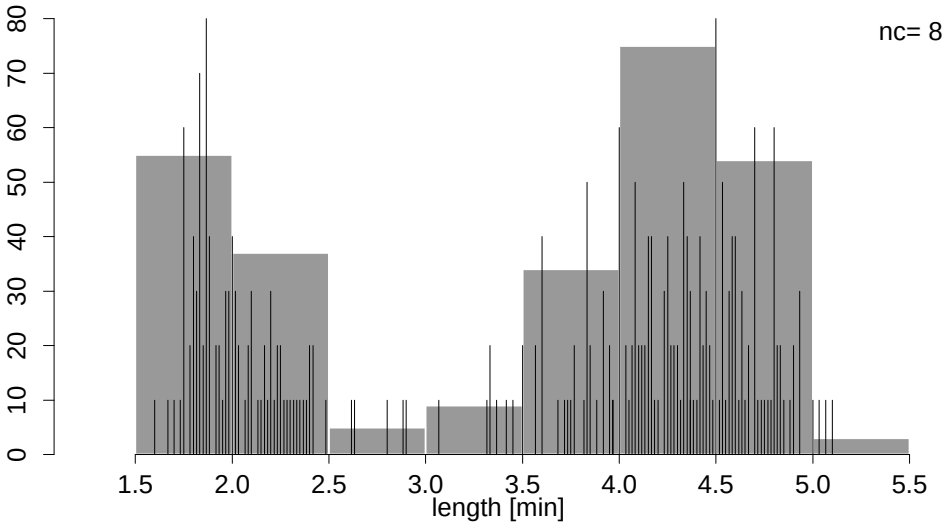
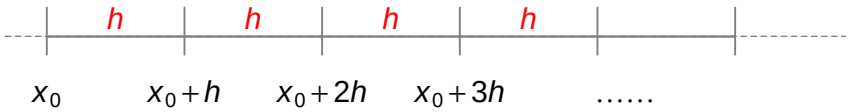


Figure 2.1: Histograms (different class widths) for durations of eruptions of "Old Faithful" geysir in Yellowstone Park.



The choice of the "origin" x_0 is highly arbitrary, whereas the role of the class width is immediately clear for the user. The form of the histogram depends very much on these two tuning parameters.

2.2.2 Kernel estimator

The naive estimator

Similar to the histogram, we can compute the relative frequency of observations falling into a small region. The density function $f(\cdot)$ at a point x can be represented as

$$f(x) = \lim_{h \to 0} \frac{1}{2h} \mathbb{P}[x - h < X \leq x + h]. \tag{2.1}$$

The naive estimator is then constructed without taking the limit in (2.1) and by replacing probabilities with relative frequencies:

$$\hat{f}(x) = \frac{1}{2hn} \#\{i; X_i \in (x - h, x + h]\}. \tag{2.2}$$

This naive estimator is only piecewise constant since every X_i is either in or out of the interval $(x - h, x + h]$. As for histograms, we also need to specify the so-called bandwidth h; but in contrast to the histogram, we do not need to specify an origin x_0.

An alternative representation of the naive estimator n (2.2) is as follows. Define the weight function

$$w(x) = \begin{cases} 1/2 & if\ |x| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} w\left(\frac{x - X_i}{h}\right).$$

If we choose instead of the rectangle weight function $w(\cdot)$ a general, typically more smooth kernel function $K(\cdot)$, we have the definition of the kernel density estimator

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right),$$
$$K(x) \geq 0, \int_{-\infty}^{\infty} K(x)dx = 1, \; K(x) = K(-x). \tag{2.3}$$

The estimator depends on the *bandwidth* $h > 0$ *which acts as a tuning parameter*. For large bandwidth h, the estimate $\hat{f}(x)$ tends to be very slowly varying as a function of x, while small bandwidths will produce a more wiggly function estimate. The positivity of the kernel function $K(\cdot)$ guarantees a positive density estimate $\hat{f}(\cdot)$ and the normalization $\int K(x)dx = 1$ implies that $\int \hat{f}(x)dx = 1$ which is necessary for $\hat{f}(\cdot)$ to be a density. Typically, the kernel function $K(\cdot)$ is chosen as a probability density which is symmetric around 0.

The smoothness of $\hat{f}(\cdot)$ is inherited from the smoothness of the kernel: if the rth derivative $K^{(r)}(x)$ exists for all x, then $\hat{f}^{(r)}(x)$ exists as well for all x (easy to verify using the chain rule for differentiation).

Popular kernels are the Gaussian kernel

$$K(x) = \varphi(x) = (2\pi)^{-\frac{1}{2}} e^{-x^2/2} \text{ (the density of the } \mathcal{N}(0,1) \text{ distribution)}$$

or a kernel with finite support such as $K(x) = \frac{\pi}{4}\cos(\frac{\pi}{2}x)I(|x| \leq 1)$. The Epanechnikov kernel, which is optimal with respect to mean squared error, is

$$K(x) = \frac{3}{4}\left(1 - |x|^2\right) I(|x| \leq 1).$$

But far more important than the kernel is the bandwidth h: its role and how to choose it are discussed below.

2.3 The role of the bandwidth

The bandwidth h is often also called the "smoothing parameter": a moment of thought will reveal that for $h \to 0$, we will have "δ-spikes" at every observation X_i, whereas $\hat{f}(\cdot)$ becomes smoother as h is increasing.

2.3.1 Variable bandwidths: k nearest neighbors

Instead of using a global bandwidth, we can use locally changing bandwidths $h(x)$.

The general idea is to use a large bandwidth for regions where the data is sparse. The k-nearest neighbor idea is to choose

$$h(x) = \text{ Euclidean distance of } x \text{ to the } k\text{th nearest observation},$$

where k is regulating the magnitude of the bandwidth. Note that generally, $\hat{f}(\cdot)$ will not be a density anymore since the integral is not necessarily equal to one.

2.3.2 The bias-variance trade-off

We can formalize the behavior of $\hat{f}(\cdot)$ when varying the bandwidth h in terms of bias and variance of the estimator. It is important to understand heuristically that

the (absolute value of the) bias of $\hat{f}$ increases and the variance of $\hat{f}$ decreases as h increases.

Therefore, if we want to minimize the mean squared error $\mathrm{MSE}(x)$ at a point x,

$$\mathrm{MSE}(x) = \mathbb{E}\left[\left(\hat{f}(x) - f(x)\right)^2\right] = \left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2 + \mathrm{Var}(\hat{f}(x)),$$

we are confronted with a **bias-variance trade-off**. As a consequence, this allows - at least conceptually - to optimize the bandwidth parameter (namely to minimize the mean squared error) in a well-defined, coherent way. Instead of optimizing the mean squared error at a point x, one may want to optimize the integrated mean squared error (IMSE)

$$\mathrm{IMSE} = \int \mathrm{MSE}(x)dx$$

which yields an integrated decomposition of squared bias and variance (integration is over the support of X). Since the integrand is non-negative, the order of integration (over the support of X and over the probability space of X) can be reversed and written as

$$\mathrm{IMSE} = \mathbb{E}[\int \left(\hat{f}(x) - f(x)\right)^2 dx]$$

which is also referred as MISE (mean integrated squared error).

2.3.3 Asymptotic bias and variance

It is straightforward (using definitions) to give an expression for the exact bias and variance:

$$\begin{aligned}
&\mathbb{E}[\hat{f}(x)] = \int \frac{1}{h} K\left(\frac{x-y}{h}\right) f(y)dy \\
&\mathrm{Var}(\hat{f}(x)) \\
&= \frac{1}{nh^2}\,\mathrm{Var}\left(K\left(\frac{x-X_i}{h}\right)\right) = \frac{1}{nh^2}\mathbb{E}\left[K\left(\frac{x-X_i}{h}\right)^2\right] - \frac{1}{nh^2}\mathbb{E}\left[K\left(\frac{x-X_i}{h}\right)\right]^2 \\
&= n^{-1}\int \frac{1}{h^2} K\left(\frac{x-y}{h}\right)^2 f(y)dy - n^{-1}\left(\int \frac{1}{h} K\left(\frac{x-y}{h}\right) f(y)dy\right)^2. \qquad (2.4)
\end{aligned}$$

For the bias we therefore get

$$\begin{aligned}
\mathrm{Bias}(x) &= \int \frac{1}{h} K\left(\frac{x-y}{h}\right) f(y)dy - f(x) \\
&\underbrace{=}_{\text{change of variable}} \int K(z) f(x-hz)dz - f(x) = \int K(z)\left(f(x-hz) - f(x)\right)dz. \qquad (2.5)
\end{aligned}$$

To approximate this expression in general, we invoke an asymptotic argument. We assume that $h \to 0$ as sample size $n \to \infty$, that is:

$$\boxed{h = h_n \to 0 \text{ with } nh_n \to \infty.}$$

This will imply that the bias goes to zero since $h_n \to 0$; the second condition requires that h_n is going to zero more slowly than $1/n$ which turns out to imply that also the variance of the estimator will go to zero as $n \to \infty$. To see this, we use a Taylor expansion of f, assuming that f is sufficiently smooth:

$$f(x - hz) = f(x) - hzf'(x) + \frac{1}{2}h^2z^2f'(x) + \dots$$

Plugging this into (2.5) yields

$$\begin{aligned}\text{Bias}(x) &= -hf'(x)\underbrace{\int zK(z)dz}_{=0} + \frac{1}{2}h^2f''(x)\int z^2K(z)dz + \dots \\ &= \frac{1}{2}h^2f''(x)\int z^2K(z)dz + \text{higher order terms in } h.\end{aligned}$$

For the variance, we get from (2.4)

$$\begin{aligned}&\text{Var}(\hat{f}(x)) = n^{-1}\int \frac{1}{h^2}K\left(\frac{x-y}{h}\right)^2 f(y)dy - n^{-1}(f(x) + \text{Bias}(x))^2 \\ &= n^{-1}h^{-1}\int f(x - hz)K(z)^2dz - \underbrace{n^{-1}(f(x) + \text{Bias}(x))^2}_{=O(n^{-1})} \\ &= n^{-1}h^{-1}\int f(x - hz)K(z)^2dz + O(n^{-1}) = n^{-1}h^{-1}f(x)\int K(z)^2dz + o(n^{-1}h^{-1}),\end{aligned}$$

assuming that f is smooth and hence $f(x - hz) \to f(x)$ as $h_n \to 0$.

In summary: for $h = h_n \to 0$, $h_n n \to \infty$ as $n \to \infty$

$$\boxed{\begin{aligned}&\text{Bias}(x) = h^2f''(x)\textstyle\int z^2K(z)dz/2 + o(h^2) \ (n \to \infty) \\ &\text{Var}(\hat{f}(x)) = (nh)^{-1}f(x)\textstyle\int K(z)^2dz + o((nh)^{-1}) \ (n \to \infty)\end{aligned}}$$

The optimal bandwidth $h = h_n$ which minimizes the leading term in the asymptotic MSE(x) can be calculated straightforwardly by solving $\frac{\partial}{\partial h}\text{MSE}(x) = 0$,

$$h_{opt}(x) = n^{-1/5}\left(\frac{f(x)\int K^2(z)dz}{(f''(x))^2(\int z^2K(z)dz)^2}\right)^{1/5}. \tag{2.6}$$

The optimal rate for MSE(x) is therefore $O(n^{-4/5})$.

2.3.4 Estimating local bandwidths

As seen from (2.6), the asymptotically best bandwidth depends on $f''(x)$ which is unknown; note that $K(\cdot)$ is known. It is possible to estimate the second derivative, again by a kernel estimator with an "initial" bandwidth h_{init} (sometimes called a pilot bandwidth) yielding $\hat{f}''_{h_{init}}(x)$. Plugging this estimate into (2.6) yields an estimated bandwidth $\hat{h}(x)$ for the density estimator $\hat{f}(\cdot)$ (the original problem): of course, $\hat{h}(x)$ depends on the initial bandwidth h_{init}, but choosing h_{init} in an ad-hoc way is less critical than choosing the bandwidth h in an ad-hoc style for the density estimator. We will show in section

3.2.1, in the related area of nonparametric regression, an example about locally changing bandwidths which are estimated based on an iterative version of the plug-in idea above.

We also see that the optimal bandwidth in (2.6) is local, depending on x. It can be important to use local bandwidths instead of a single global bandwidth in a kernel estimator; at the expense of being more difficult to determine. But the plug-in procedure outlined above describes a methodology how to estimate local bandwidths from data and hence how to implement a kernel estimator with locally varying bandwidths.

2.4 Higher dimensions

Quite many applications involve multivariate data. For simplicity, consider data which are i.i.d. realizations of d-dimensional random variables

$$\mathbf{X}_1, \ldots, \mathbf{X}_n \text{ i.i.d. } \sim f(x_1, \ldots, x_d)dx_1 \cdots dx_d$$

where $f(\cdot)$ denotes the multivariate density.

The multivariate kernel density estimator is, in its simplest form, defined as

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right),$$

where the kernel $K(\cdot)$ is now a function, defined for d-dimensional $\mathbf{x}$, satisfying

$$K(\mathbf{x}) \geq 0, \int_{\mathbb{R}^d} K(\mathbf{x})d\mathbf{x} = 1, \ K(\cdot) \text{ radially symmetric.}$$

Usually, the kernel $K(\cdot)$ is chosen as a product of a kernel K_{univ} for univariate density estimation

$$K(\mathbf{u}) = \prod_{j=1}^{d} K_{univ}(u_j).$$

2.4.1 The curse of dimensionality

In practice, multivariate kernel density estimation is often restricted to dimension $d = 2$. The reason is, that a higher dimensional space (with d of medium size or large) will be only very sparsely populated by data points. Or in other words, there will be only very few neighboring data points to any value $\mathbf{x}$ in a higher dimensional space, unless the sample size is extremely large. This phenomenon is also called the *curse of dimensionality*.

An implication of the curse of dimensionality is the following lower bound for the best mean squared error of nonparametric density estimators (assuming that the underlying density is twice differentiable): it has been shown that the best possible MSE rate is

$$O(n^{-4/(4+d)}).$$

The following table evaluates $n^{-4/(4+d)}$ for various n and d:

$n^{-4/(4+d)}$	$d = 1$	$d = 2$	$d = 3$	$d = 5$	$d = 10$
$n = 100$	0.025	0.046	0.072	0.129	0.268
$n = 1000$	0.004	0.010	0.019	0.046	0.139
$n = 100'000$	$1.0\ 10^{-4}$	$4.6\ 10^{-4}$	$13.9\ 10^{-4}$	0.006	0.037

Thus, to for $d = 10$, the rate with $n = 100'000$ is still 1.5 times worse than for $d = 1$ and $n = 100$.

Chapter 3

Nonparametric Regression

3.1 Introduction

We consider here nonparametric regression with one predictor variable. Practically relevant generalizations to more than one or two predictor variables are not so easy due to the curse of dimensionality mentioned in section 2.4.1 and often require different approaches, as will be discussed later in Chapter - "under construction" -.

Figure 3.1 shows (several identical) scatter plots of (x_i, Y_i) $(i = 1, \ldots, n)$. We can model such data as

$$Y_i = m(x_i) + \varepsilon_i, \tag{3.1}$$

where $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d. with $\mathbb{E}[\varepsilon_i] = 0$ and $m : \mathbb{R} \to \mathbb{R}$ is an "arbitrary" function. The function $m(\cdot)$ is called the nonparametric regression function and it satisfies $m(x) = \mathbb{E}[Y|x]$. The restriction we make for $m(\cdot)$ is that it fulfills some kind of smoothness conditions. The regression function in Figure 3.1 does nor appear to be linear in x and linear regression is not a good model. The flexibility to allow for an "arbitrary" regression function is very desirable; but of course, such flexibility has its price, namely an inferior estimation accuracy than for linear regression.

3.2 The kernel regression estimator

We can view the regression function in (3.1) as

$$m(x) = \mathbb{E}[Y|X = x],$$

(assuming that X is random and $X_i = x_i$ are realized values of the random variables). We can express this conditional expectation as

$$\int_{\mathbb{R}} y f_{Y|X}(y|x)dy = \frac{\int_{\mathbb{R}} y f_{X,Y}(x,y)dy}{f_X(x)},$$

where $f_{Y|X}$, $f_{X,Y}$, f_X denote the conditional, joint and marginal densities. We can now plug in the univariate and bivariate kernel density (all with the same univariate kernel K) estimates

$$\hat{f}_X(x) = \frac{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)}{nh}, \quad \hat{f}_{X,Y}(x,y) = \frac{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) K\left(\frac{y-Y_i}{h}\right)}{nh^2}$$

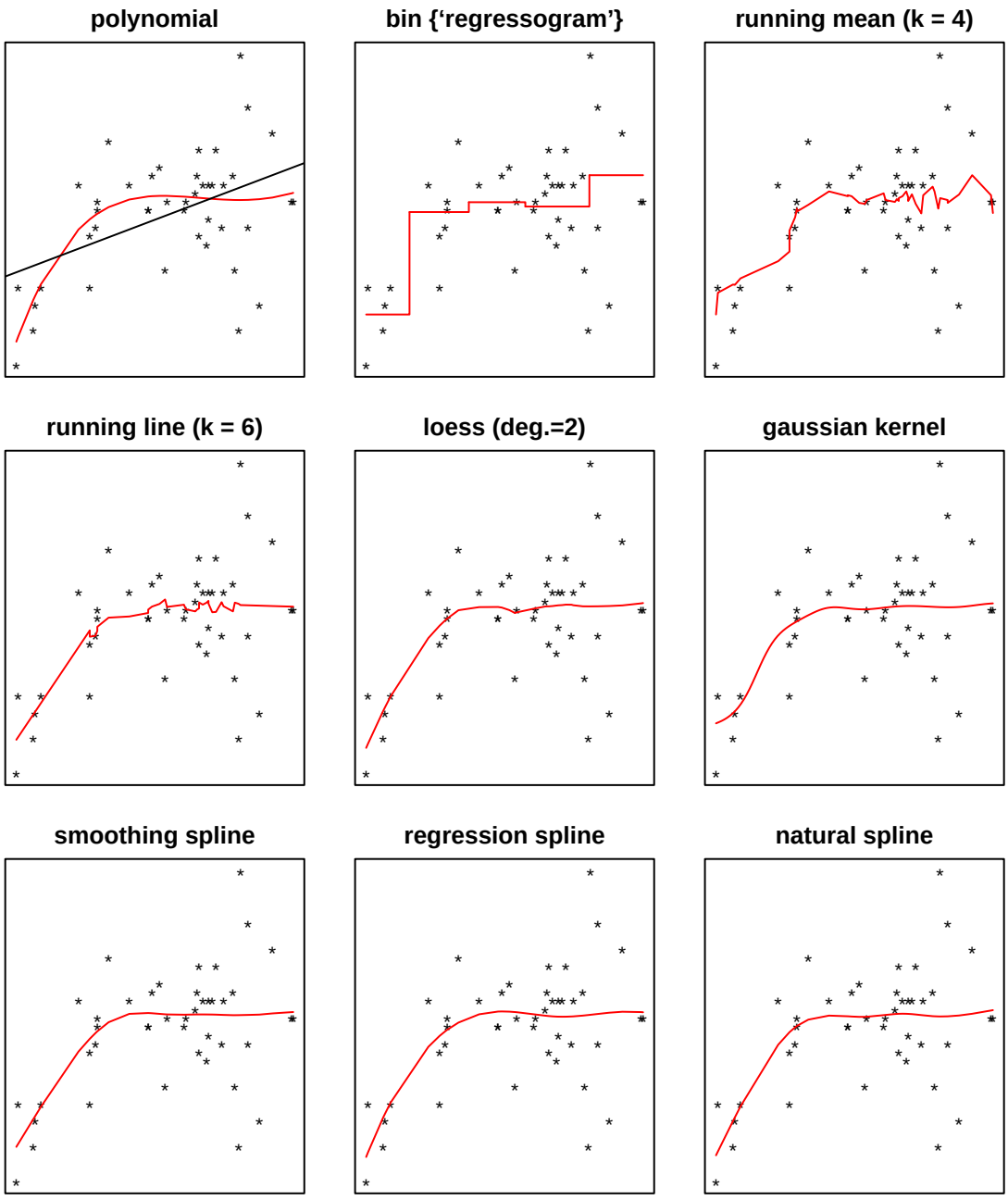


Figure 3.1: Various regression estimators in model $Y_i = m(x_i) + \varepsilon_i$ $(i = 1, \ldots, 43)$ with response Y a log-concentration of a serum (in connection of Diabetes) and predictor variable x the age in months of children. See Hastie and Tibshirani (1990, p.10). Except for the linear regression fit (top left panel), all other estimators have about 5 degrees of freedom.

into the formula above which yields the so-called Nadaraya-Watson kernel estimator

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K((x - x_i)/h) Y_i}{\sum_{i=1}^{n} K((x - x_i)/h)}. \tag{3.2}$$

An interesting interpretation of the kernel regression estimator in (3.2) is

$$\hat{m}(x) = \arg\min_{m_x \in \mathbb{R}} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)(Y_i - m_x)^2. \tag{3.3}$$

This can be easily verified by solving $\frac{d}{dm_x}\sum_{i=1}^{n} K((x-x_i)/h)(Y_i - m_x)^2 = 0$. Thus, for every fixed x, we are searching for the best local constant m_x such that the localized sum of squares is minimized; localization is here described by the kernel and gives a large weight to observations (x_i, Y_i) whenever x_i is close to the point x of interest.

3.2.1 The role of the bandwidth

Analogously as in section 2.3, the bandwidth h controls the bias-variance trade-off: a large bandwidth h implies high bias but small variance, resulting in a slowly varying curve, and vice-versa.

Local bandwidth selection

Similarly as in (2.6), there is a formula of the asymptotically best local bandwidth $h_{opt}(x)$ which depends on $m''(\cdot)$ and the error variance σ_ε^2:

$$h_{opt}(x) = n^{-1/5}\left(\frac{\sigma_\varepsilon^2 \int K^2(z)dz}{\{m''(x)\int z^2 K(z)dz\}^2}\right)^{1/5}.$$

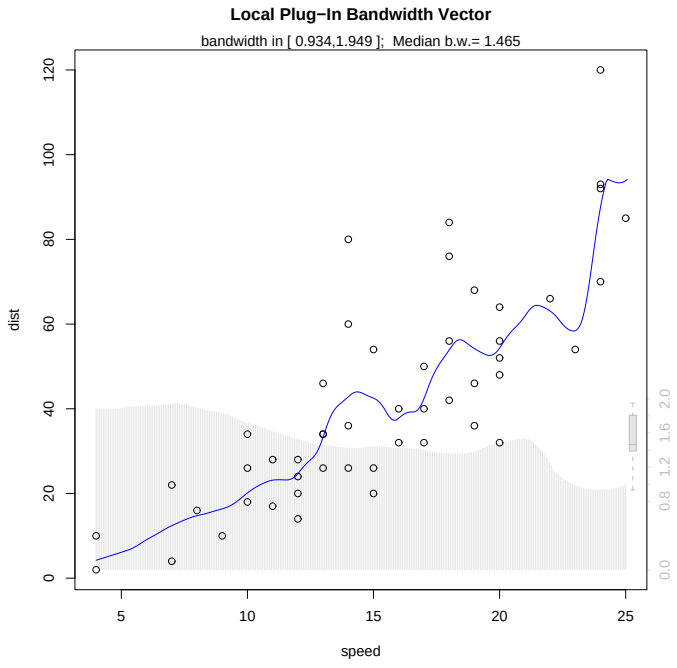


Figure 3.2: Nonparametric function estimate and locally varying bandwidths for distance of stopping as a function of speed of cars.

The locally optimal bandwidth $h_{opt}(x)$ can then be estimated in an iterative way using the plug-in principle. Roughly speaking, start with an initial bandwidth h_0 to estimate

$m''(\cdot)$ (by using an inflated version $n^{1/10}h_0$) and σ_ε^2; these estimates can then be used to get a first estimate of $h_{opt}(x)$. Now use this first bandwidth estimate as the current bandwidth h_1 to estimate again $m''(\cdot)$ (by using the inflated version $n^{1/10}h_1$) and σ_ε^2, and then obtain new estimates for $h_{opt}(x)$; and so on, see Brockmann et al. (1993).

Such a procedure has been implemented in `R` with the function `lokerns` in the library `lokern`. The dataset `cars` contains the distance for stopping as a function of speed of a car. A nonparametric function estimate with locally varying bandwidth can then be obtained as follows:

```
library(lokern)
data(cars)
lofit <- lokerns(cars$ speed, cars$ dist)
```

3.2.2 Inference for the underlying regression curve

We consider here the properties, in particular the variability, of the kernel estimator $\hat{m}(x_i)$ at an observed design point x_i.

The hat matrix

It is useful to represent the kernel estimator evaluated at the design points $\hat{m}(x_1), \ldots, \hat{m}(x_n)$ as a *linear* operator (a matrix):

$$\begin{aligned} \mathcal{S}: \quad & \mathbb{R}^n \to \mathbb{R}^n, \\ & (Y_1, \ldots, Y_n)^T \mapsto (\hat{m}(x_1), \ldots, \hat{m}(x_n))^T. \end{aligned}$$

The kernel estimator in (3.2) is of the form

$$\hat{m}(x) = \sum_{i=1}^n w_i(x) Y_i, \;\; w_i(x) = \frac{K((x - x_i)/h)}{\sum_{j=1}^n K((x - x_j)/h)}.$$

Therefore, the matrix $\mathcal{S}$, which represents the operator above, is

$$[\mathcal{S}]_{r,s} = w_s(x_r), \;\; r, s \in \{1, \ldots, n\},$$

since $\mathcal{S}\{(Y_1, \ldots, Y_n)^T\} = (\hat{m}(x_1), \ldots, \hat{m}(x_n))^T$. The matrix $\mathcal{S}$ is sometimes referred to the hat matrix, yielding the vector of fitted values (at the observed design points x_i). An elementary formula then yields for the covariances

$$\mathrm{Cov}(\hat{m}(x_i), \hat{m}(x_j)) = \sigma_\varepsilon^2 (\mathcal{S}\mathcal{S}^T)_{ij},$$

and

$$\mathrm{Var}(\hat{m}(x_i)) = \sigma_\varepsilon^2 (\mathcal{S}\mathcal{S}^T)_{ii}. \tag{3.4}$$

Degrees of freedom

One way to assign degrees of freedom for regression estimators with a linear hat-operator $\mathcal{S}$ is given by

$$df = \mathrm{trace}(\mathcal{S}).$$

This definition coincides with the notion in the linear model: there, the fitted values $\hat{Y}_1, \ldots, \hat{Y}_n$ can be represented by the projection $P = X(X^TX)^{-1}X^T$, which is the hat matrix, and $\mathrm{trace}(P) = \mathrm{trace}((X^TX)^{-1}X^TX) = \mathrm{trace}(I_p) = p$ equals the number of parameters in the model. Thus, the definition of degrees of freedom above can be viewed as a general concept for the number of parameters in a model fit with linear hat matrix.

Estimation of the error variance

Formula (3.4) requires knowledge of σ_ε^2. A plausible estimate is via the residual sum of squares,

$$\hat{\sigma}_\varepsilon^2 = \frac{\sum_{i=1}^n (Y_i - \hat{m}(x_i))^2}{n - df}.$$

We then get an estimate for the standard error of the kernel regression estimator at the design points via (3.4):

$$\widehat{s.e}(\hat{m}(x_i)) = \sqrt{\widehat{\mathrm{Var}}(\hat{m}(x_i))} = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{m}(x_i))^2}{n - df} (\mathcal{S}\mathcal{S}^T)_{ii}}.$$

The estimated standard errors above are useful since under regularity conditions, $\hat{m}(x_i)$ is asymptotically normal distributed:

$$\hat{m}(x_i) \approx \mathcal{N}(\mathbb{E}[\hat{m}(x_i)], \mathrm{Var}(\hat{m}(x_i))),$$

so that

$$I = \hat{m}(x_i) \pm 1.96 \widehat{s.e.}(\hat{m}(x_i))$$

yields approximate pointwise confidence intervals for $\mathbb{E}[\hat{m}(x_i)]$. Some functions in `R` (e.g. the function `gam` for the library `mgcv`, see Chapter - "under construction"-) supply such pointwise confidence intervals. Unfortunately, it is only a confidence interval for the expected value $\mathbb{E}[\hat{m}(x_i)]$ and not for the true underlying function $m(x_i)$. Correction of this interval is possible by subtracting a bias estimate: i.e., instead of the interval I above, we can use

$$I - \widehat{\mathrm{bias}},$$

where $\widehat{\mathrm{bias}}$ is an estimate of the bias (which is not so easy to construct; see also section 2.3).

3.3 Local polynomial nonparametric regression estimator

As a starting point, consider the kernel estimator which can be represented as a locally constant function as in (3.3). This can now be extended to functions which are locally polynomial. We aim to find local regression parameters $\beta(x)$, defined as

$$\hat{\beta}(x) = \underset{\beta \in \mathbb{R}^p}{\arg\min} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \left(Y_i - \beta_1 - \beta_2(x_i - x) - \ldots - \beta_p(x_i - x)^{p-1}\right)^2.$$

An even number p turns out to be better: in practice, we often choose $p = 2$ or $p = 4$. The estimated local regression parameter $\hat{\beta}(x)$ describes a local polynomial regression fit, localized and centered at x. The function estimator is then given by evaluating this local regression fit $\sum_{j=1}^p \hat{\beta}_j(x)(u-x)^{j-1}$ at $u = x$: due to the centering, only the local intercept remains and the local polynomial function estimator becomes

$$\hat{m}(x) = \hat{\beta}_1(x).$$

Note that due to (local) correlation among the $(x_i - x)^j$'s, $\hat{\beta}_1(x)$ is not the same as a local constant fit from (3.3).

The local polynomial estimator is often better at the edges than the locally constant Nadaraya-Watson kernel estimator. Another interesting property is that the method also immediately yields estimates for the derivatives of the function: when differentiating the local regression fit $\sum_{j=1}^{p} \hat{\beta}_j(x)(u-x)^{j-1}$ with respect to u and evaluating it at x, we obtain

$$\hat{m}^{(r)}(x) = r!\hat{\beta}_{r+1}(x) \ (r = 0, 1, \ldots p-1).$$

3.4 Smoothing splines and penalized regression

Function estimation could also be done by using higher order global polynomials, which is often not advisable, or by using splines which can be specified by choosing a set of knots. The latter is a more locally oriented approach and is called "regression splines". Here, we discuss a method based on splines *without* having to specify where to select the knots of the spline.

3.4.1 Penalized sum of squares

Consider the following problem: among all functions f with continuous second derivatives, find the one which minimizes the penalized residual sum of squares

$$\sum_{i=1}^{n} (Y_i - m(x_i))^2 + \lambda \int m''(z)^2 dz, \tag{3.5}$$

where $\lambda \geq 0$ is a smoothing parameter. The first term measures closeness to the data and the second term penalizes curvature ("roughness") of the function. The two extreme cases are:

- $\lambda = 0$: m is any function interpolating the data;
- $\lambda = \infty$: the least squares fit for linear regression since no second derivative can be tolerated.

Thus, a large λ corresponds to a smooth function.

3.4.2 The smoothing spline solution

Remarkably, the minimizer of (3.5) is finite-dimensional, although the criterion to be minimized is over a Sobolev space of functions (function space for which the integral $\int m''^2$ is defined), an infinite-dimensional space.

The solution $\hat{m}_\lambda(\cdot)$ (e.g. the minimizer of (3.5)) is a natural **cubic spline** with knots at the predictors x_i: that is, $\hat{m}$ is a piecewise cubic polynomial in each interval $[x_i, x_{i+1})$ such that $\hat{m}_\lambda^{(k)}$ $(k = 0, 1)$ is continuous everywhere.

Knowing that the solution is a cubic spline, the solution can be obtained by linear algebra. The trick is to represent

$$\hat{m}_\lambda(x) = \sum_{j=1}^{n} \beta_j B_j(x), \tag{3.6}$$

where the $B_j(\cdot)$'s are basis functions for natural splines. The unknown coefficients can then be estimated from least squares in linear regression under side constraints. The criterion in (3.5) for $\hat{m}_\lambda$ as in (3.6) then becomes

$$\|Y - B\beta\|^2 + \lambda\beta^T\Omega\beta,$$

where the design matrix B has jth column $(B_j(x_1), \ldots, B_j(x_n))^T$ and $\Omega_{jk} = \int B_j''(z)B_k''(z)dz$. The solution can then be derived in a straightforward way,

$$\hat{\beta}_{n\times 1} = (B^TB + \lambda\Omega)^{-1}B^TY. \tag{3.7}$$

This can be computed efficiently using fast linear algebra.

The fitted values are then $\hat{Y}_i = \hat{m}_\lambda(x_i)$ $(i = 1, \ldots, n)$ and

$$(\hat{Y}_1, \ldots, \hat{Y}_n)^T = \mathcal{S}_\lambda Y, \quad \mathcal{S}_\lambda = B(B^TB + \lambda\Omega)^{-1}B^T.$$

The hat matrix $\mathcal{S}_\lambda = \mathcal{S}_\lambda^T$ is here symmetric which implies elegant mathematical properties (real-valued eigen-decomposition).

3.4.3 Shrinking towards zero

At first sight, the smoothing spline solution in (3.6) looks heavily over-parameterized since we have to fit n unknown coefficients $\beta_1, \ldots, \beta_n$. However, the solution in (3.7) is not the least squares estimator but rather a Ridge-type version: the matrix $\lambda\Omega$ serves as a Ridge or shrinkage matrix so that the estimates $\hat{\beta}$ are shrunken towards zero: i.e. for large λ, the expression $(B^TB + \lambda\Omega)^{-1}$ becomes small. Thus, since all the coefficients are shrunken towards zero, we gain on the variance part of each $\hat{\beta}_j$ by the square of the shrinkage factor, and the overall smoothing spline fit will be appropriate if λ is chosen suitably.

Note that λ can be chosen on the scale of equivalent degrees of freedom (df): df = $trace(\mathcal{S}_\lambda)$. This provides an intuitive way to specify a smoothing spline: e.g. a smoothing spline with df=5 is as complex as a global polynomial of degree 4 (which has 5 parameters including the intercept), see also Figure 3.1.

3.4.4 Relation to equivalent kernels

It is interesting to note that there is a relationship between the smoothing spline estimate and a particular kernel estimator. The smoothing spline estimate $\hat{m}(x)$ is approximately

$$\begin{aligned}
\hat{m}_\lambda(x) &\approx \sum_{i=1}^n w_i(x)Y_i, \\
w_i(x) &= \frac{1}{nh(x)f_X(x)}K\left(\frac{x - x_i}{h(x)}\right), \\
h(x) &= \lambda^{1/4}n^{-1/4}f_X(x)^{-1/4}, \\
K(u) &= \frac{1}{2}\exp\left(-\frac{|u|}{\sqrt{2}}\right)\sin\left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4}\right).
\end{aligned}$$

See for example Green and Silverman (1994, Ch. 3.7).

The important fact is here that the bandwidth of the equivalent kernel estimator has a *local bandwidth*, depending on the density of the predictor variable x. In regions where the density of the predictor is low (observations are sparse), the bandwidth automatically

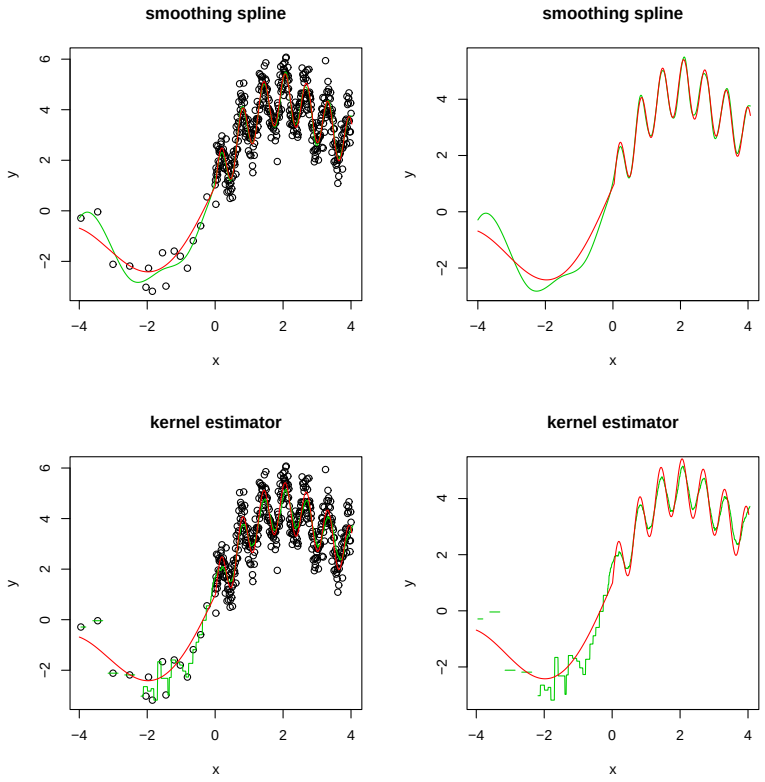


Figure 3.3: Curve estimation for synthetic dataset. Left panels: scatterplot, overlaid with curve estimates (green) and true curve (red); Right panels: curve estimates (green) and true curve (red). Top: Smoothing spline with GCV-selected df; Bottom: Nadaraya-Watson kernel estimator with bandwidth chosen for good estimation of the strong oscillations of the true function. Sample size is $n = 417$.

adapts and becomes large: intuitively, this is the right behavior because we should use strong smoothing in regions where only few observations are available.

An example of a smoothing spline fit for real data is displayed in Figure 3.1. Finally, we illustrate on an artificial dataset the advantage of smoothing splines to adapt to the density of the predictor variables. Figure 3.3 shows the performance of smoothing splines in comparison with the Nadaraya-Watson Gaussian kernel estimator. The data has the following structure:

- the density of the predictor is high for positive values and low for negative values
- the true function is strongly oscillating where the predictor density is high and slowly oscillating where the predictors are sparse

The smoothing spline fit (using the GCV criterion for selecting the degrees of freedom, see section 4.5) yields a very good fit: it captures the strong oscillations because there are many data points with positive values of the predictors. On the other hand, the kernel estimator has been tuned such that it captures the strong oscillations, using a small bandwidth h (this was done by knowing the true underlying function – which is not feasible in practice): but the small bandwidth h then causes a much too rough and poor estimate for negative predictor values, although the underlying true function is smooth.

Chapter 4

Cross-Validation

4.1 Introduction

The generalization performance (which is a terminology from the machine learning community) of a learning method describes its prediction capacity on new test data. Or in other words, the generalization performance measures the predictive power of a learning method on new, out-sample data. Having an estimate of this predictive power is very important for comparing among different methods or for tuning an estimator or algorithm to optimize predictive power.

Cross-Validation is one very general method for estimating such generalization or out-sample performance.

4.2 Training and Test Set

Consider the situation where we have data

$$(X_1, Y_1), \ldots, (X_n, Y_n) \text{ i.i.d. } \sim P$$

where P denotes the unknown distribution. (In case where the x_i is non-random, we can also drop the i.i.d. assumption for the pairs (x_i, Y_i)).

We typically have a target in mind, for example the regression function $m(\cdot)$. The estimated regression function $\hat{m}(\cdot)$ is then constructed by using some estimator or algorithm, based on the data $(X_1, Y_1), \ldots, (X_n, Y_n)$. This data is also called the **training data**, since it is used to train the estimator or learning algorithm.

We then would like to evaluate the accuracy of the estimated target which is based on the training data. A principal problem thereby is that if we use the training data again to measure the predictive power of our estimated target (e.g. of $\hat{m}(\cdot)$), the results will be overly optimistic. For example, when using the residual sum or squares in regression,

$$\sum_{i=1}^{n} (Y_i - \hat{m}(X_i))^2$$

becomes smaller the more "complex" (more degrees of freedom) the model for $\hat{m}(\cdot)$ involves. Obviously, a very complex model will not be good. We have seen already in Section 3.4 that penalizing the residual sum of squares can be useful to cope with the overfitting problem.

Alternatively, we could look how well the estimated target, e.g. $\hat{m}(\cdot)$, does on **new test data**

$$(X_{new,1}, Y_{new,1}), \ldots, (X_{new,\ell}, Y_{new,\ell}) \text{ i.i.d. } \sim P,$$

which is assumed to be independent from the training data but having the same distribution P. In case of regression, we would like to evaluate

$$\ell^{-1} \sum_{i=1}^{\ell} (Y_{new,i} - \hat{m}(X_{new,i}))^2.$$

More generally, when having an estimated target $\hat{m}$ and a loss function ρ, we would like to evaluate

$$\ell^{-1} \sum_{i=1}^{\ell} \rho\left(Y_{new,i}, \hat{m}(X_{new,i})\right),$$

where $\hat{m}(\cdot)$ is constructed from training data only. If ℓ is large, this evaluation on the test set approximates the theoretical test set error

$$\mathbb{E}_{(X_{new},Y_{new})}[\rho(Y_{new}, \underbrace{\hat{m}}_{\text{based on training data only}}(X_{new}))]$$

which is still a function of the training data. The expected value of this (with respect to the training data) is the eneralization error,

$$\mathbb{E}_{training}\mathbb{E}_{(X_{new},Y_{new})}[\rho(Y_{new}, \underbrace{\hat{m}}_{\text{based on training data only}}(X_{new}))] = \mathbb{E}[\rho(Y_{new}, \hat{m}(X_{new}))], \quad (4.1)$$

where the latter $\mathbb{E}$ is over the training data $(X_1, Y_1), \ldots, (X_n, Y_n)$ as well as the test data (X_{new}, Y_{new}). This generalization error avoids the fact that it would get smaller the more complex the model for $\hat{m}(\cdot)$. In particular, it will increase as $\hat{m}$ is overfitting. Hence the generalization error in (4.1) is a very useful quantity to regularize and tune an estimator or algorithm for $\hat{m}$.

4.3 Constructing training-, test-data and cross-validation

Unfortunately, test data is not available at the time we want to run and tune our estimator or algorithm. But we can always "artificially" construct (smaller) training- and test-data.

4.3.1 Leave-one-out cross-validation

For leave-one-out cross-validation (CV), we use the ith sample point as test data (test sample size $= 1$) and the remaining $n - 1$ sample points as training data.

Denote in general the estimator or algorithm by $\hat{\theta}_n$ which is based on the n sample points. In CV, when deleting the ith sample, the estimator is based on the sample without the ith observation, and we denote this by

$$\hat{\theta}_{n-1}^{(-i)}, \ i = 1, \ldots, n.$$

We can then evaluate this estimate on the ith observation (the test sample), for every $i = 1, \ldots, n$. To make this more explicit, we suppose that the estimator $\hat{\theta}$ is a curve

estimator $\hat{m}$, and performance is evaluated in terms of a loss function ρ, e.g. $\rho(u) = u^2$ as in Chapter 3. The cross-validated performance is then

$$n^{-1} \sum_{i=1}^{n} \rho \left(Y_i, \hat{m}_{n-1}^{(-i)}(X_i) \right) \tag{4.2}$$

which is an estimate of the test set error, or generalization error, in (4.1).

Note that the CV-method is very general: it can be used for any loss function ρ and in many problems which can be different than linear or nonparametric regression.

CV in general requires that the estimator $\hat{\theta}$ is fitted n-times, namely for all training sets where the ith observation has been deleted ($i = 1, \ldots, n$).

4.3.2 K-fold Cross-Validation

A computationally cheaper version of leave-one-out CV is the so-called K-fold CV. The data set is randomly partitioned into K equally sized (as equal as possible) subsets $\mathcal{B}_k$ of $\{1, \ldots, n\}$ such that $\cup_{k=1}^{K} \mathcal{B}_k = \{1, \ldots, n\}$ and $\mathcal{B}_j \cap \mathcal{B}_k = \emptyset$ ($j \neq k$). We can now set aside a kth test data set including all sample points whose indices are elements of $\mathcal{B}_k$.

K-fold cross-validation then uses the sample points with indices not in $\mathcal{B}_k$ as training set to construct an estimator

$$\hat{\theta}_{n-|\mathcal{B}_k|}^{(-\mathcal{B}_k)}.$$

The analogue of the evaluation formula in (4.2) is then, for regression with $\hat{m}(\cdot)$,

$$K^{-1} \sum_{k=1}^{K} |\mathcal{B}_k|^{-1} \sum_{i \in \mathcal{B}_k} \rho \left(Y_i, \hat{m}_{n-|\mathcal{B}_k|}^{-(\mathcal{B}_k)}(X_i) \right).$$

K-fold CV thus only needs to run the estimation algorithm K times. Leave-one-out CV is the same as n-fold CV. In practice, often $K = 5$ or $K = 10$ are used.

4.3.3 Random divisions into test- and training-data

K-fold CV has the disadvantage that it depends on the **one** realized random partition into subsets $\mathcal{B}_1, \ldots, \mathcal{B}_K$. In particular, if the data (pairs) are assumed to be i.i.d., the indexing of the data (pairs) should not have an influence on validating a performance: note that leave-one-put CV is indeed independent of indexing the data.

In principle, we can generalize leave-one-out CV to leave-d-out CV. Leave a set $\mathcal{C}$ comprising d observations out (set them aside as test data) and use the remaining $n - d$ data points for training the statistical model or algorithm:

$$\hat{\theta}_{n-d}^{(-\mathcal{C})}, \text{ for all possible subsets } \mathcal{C}_k, \ k = 1, 2, \ldots, \binom{n}{d}.$$

We can then evaluate this estimate on observations from the test set $\mathcal{C}_i$ (the test sample), for every i. The analogue of (4.2), for regression with $\hat{m}(\cdot)$, is then

$$\binom{n}{d}^{-1} \sum_{k=1}^{\binom{n}{d}} d^{-1} \sum_{i \in \mathcal{C}_k} \rho \left(Y_i, \hat{m}_{n-d}^{(-\mathcal{C}_k)}(X_i) \right) \tag{4.3}$$

which is also an estimate of the test set error, or generalization error in (4.1).

The computational burden becomes immense if $d \geq 3$. A computational short-cut is then given by randomization: instead of considering *all* subsets (test sets), we draw B **random** test subsets

$$\mathcal{C}_1^*, \ldots, \mathcal{C}_B^* \text{ i.i.d. } \sim \text{Uniform}(\{1, \ldots, \binom{n}{d}\}),$$

where the Uniform distribution assigns probability $\binom{n}{d}^{-1}$ to every possible subset of size d. Such a Uniform distribution, or such a random subset $\mathcal{C}^*$, is easily constructed by **sampling without replacement**:

draw d times randomly without replacement from $\{1, \ldots, n\}$, yielding a subset $\mathcal{C}^*$.

The random approximation for (4.3) is then

$$B^{-1} \sum_{k=1}^{B} d^{-1} \sum_{i \in \mathcal{C}_k^*} \rho\left(Y_i, \hat{m}_{n-d}^{(-\mathcal{C}_k^*)}(X_i)\right). \tag{4.4}$$

For $B = \infty$, the two expressions in (4.4) and (4.3) coincide (note that we only would need a finite, maybe huge, amount of computation for evaluation of (4.3)).

In practice, we typically choose $d = [\gamma n]$ with $\gamma \approx 0.1$ (10% test data). For the number of random test and training sets, we choose $B \approx 50 - 500$, depending on the cost to compute $\hat{\theta}_{n-d}^{(-\mathcal{C})}$ for a training set of size $n - d$ (evaluation on the test set $\mathcal{C}$) is usually fast). Thus, in terms of computation, the stochastic version in (4.4) may be even faster than leave-one-out CV in (4.2) if $B < n$, and we can use the stochastic approximation also for leave-one-out CV when sample size n is large.

4.4 Properties of different CV-schemes

4.4.1 Leave-one-out CV

Leave-one-out CV, which is equal to n-fold CV, is approximately unbiased for the true prediction or generalization error: the only drawback in terms of bias is that we use training sample size $n-1$ instead of the original n, causing for a slight bias. The variance of leave-one-out CV is typically high, because the n training sets are so similar to each other:

$$\text{Var}\left(n^{-1} \sum_{i=-1}^{n} \rho\left(Y_i, \hat{m}_{n-1}^{(-i)}(X_i)\right)\right) = n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \text{Cov}\left(\rho(Y_i, \hat{m}_{n-1}^{(-i)}(X_i)), \rho(Y_j, \hat{m}_{n-1}^{(-j)}(X_j))\right).$$

Although (X_i, Y_i) is typically assumed to be independent from (X_j, Y_j), the covariances are substantial because of strong correlation of $\hat{m}_{n-1}^{(-i)}(\cdot)$ and $\hat{m}_{n-1}^{(-j)}(\cdot)$, and hence the double sum in the formula above can be quite large.

4.4.2 Leave-d-out CV

Heuristically, it is clear that leave-d-out CV, with $d > 1$, has higher bias than leave-one-out CV; because we use training samples of sizes $n - d$ instead of the original n, causing some bias. In terms of variance, we average over more, although highly correlated summands in (4.3), which can be shown to decrease variance in comparison to leave-one-out CV.

4.4.3 Versions with computational shortcuts

K-fold CV has larger bias than leave-one-out CV; because the training sets are of smaller sample size than $n-1$ (in leave-one-out CV), and also smaller than the original sample size n. In terms of variance, it is not clear (sometimes wrongly stated in the literature) whether K-fold CV has smaller variance than leave-one-out CV.

The stochastic approximation is expected to have somewhat higher bias and variance than the computationally infeasible leave-d-out CV: it is difficult to assess how much we lose by using a finite B.

4.5 Computational shortcut for some linear fitting operators

Consider the special case of fitting a cubic smoothing spline or fitting a least squares parametric estimator: in both cases, we have a linear fitting operator $\mathcal{S}$,

$$(\hat{m}(x_1), \ldots, \hat{m}(x_n))^T = \mathcal{S}\mathbf{Y}, \ \mathbf{Y} = (Y_1, \ldots, Y_n)^T.$$

When focusing on the squared error loss function $\rho(y,x) = |y-x|^2$, there is a surprising result for representing the leave-one-out CV score in (4.2):

$$n^{-1}\sum_{i=1}^{n}\left(Y_i - \hat{m}_{n-1}^{(-i)}(X_i)\right)^2 = n^{-1}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{m}(X_i)}{1-\mathcal{S}_{ii}}\right)^2. \tag{4.5}$$

The interesting property is that we can compute the CV score by fitting the original estimator $\hat{m}(\cdot)$ **once on the full dataset**, without having to do it n times by holding one observation back as a test point. Moreover, by using efficient linear algebra implementations, the elements $\mathcal{S}_{ii}$ can be computed with $O(n)$ operations.

Historically, it was computationally more difficult to obtain all diagonal elements from the hat matrix $\mathcal{S}_{ii}$ $(i = 1, \ldots, n)$; and computing the trace$(\mathcal{S})$, which equals the sum of the eigenvalues of $\mathcal{S}$, has been easier. The generalized cross-validation was then proposed in the late 70's:

$$\text{GCV} = \frac{n^{-1}\sum_{i=1}^{n}(Y_i - \hat{m}(X_i))^2}{(1-n^{-1}\text{trace}(\mathcal{S}))^2}.$$

This again requires to compute $\hat{m}(\cdot)$ only once on the full dataset. Moreover, if all the diagonal elements $\mathcal{S}_{ii}$ would be equal (which is typically not the case), GCV would coincide with the formula in (4.5). As outlined already, GCV has been motivated by computational considerations which are nowadays not very relevant anymore. However, the statistical properties of GCV can also be as good (and sometimes better or worse) than the ones from leave-one-out CV.

Chapter 5

Bootstrap

5.1 Introduction

The bootstrap, proposed by Efron (1979), is by now considered as a breakthrough in statistics. Essentially, the bootstrap can be described as "simulating from an estimated model". This turns out to be tremendously useful for making statistical inference (confidence intervals and testing) and, analogous to the goals in cross-validation, for estimating the predictive power of a model or algorithm and hence also for tuning of statistical procedures.

5.2 Efron's nonparametric bootstrap

Consider the situation where the data are realizations of

$$Z_1, \ldots, Z_n \text{ i.i.d. } \sim P,$$

where P denotes an unknown distribution. The variables Z_i can be real-valued (usually then denoted by X_i), or they can be vector-valued. For example, $Z_i = (X_i, Y_i)$ are the pairs in a regression problem with $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$.

We denote a statistical procedure or estimator by

$$\hat{\theta}_n = g(Z_1, \ldots, Z_n) \tag{5.1}$$

which is a (known) function g of the data $Z_1, \ldots, Z_n$. The estimator $\hat{\theta}_n$ can be a parameter estimator or also a curve estimator (e.g. in nonparametric regression).

Whenever we want to make statistical inference, we would like to know the probability distribution of $\hat{\theta}_n$. For example, constructing a confidence interval for a true parameter θ requires the knowledge of the distribution of $\hat{\theta}_n$; or constructing a statistical test requires the distribution of $\hat{\theta}_n$ under the null-hypothesis. We also considered in chapter 4 the problem of estimating the generalization error in regression

$$\mathbb{E}[(Y_{new} - \hat{m}(X_{new}))^2]$$

which can be thought of as the expected value, a summary statistic of the distribution, of

$$\hat{\theta}_{n+1} = g(Z_1, \ldots, Z_n, Z_{new}) = (Y_{new} - \hat{m}_{Z_1,\ldots,Z_n}(X_{new}))^2, \ Z_i = (X_i, Y_i),$$

where we have a function g of the training and test data.

Deriving the exact distribution of $\hat{\theta}_n$ is typically impossible, unless the function g is simple and P is a mathematically convenient distribution, e.g. P is a Normal distribution. If exact distributions are not available, much mathematical theory has been developed to get at least the asymptotic distribution of $\hat{\theta}_n$ as n gets large. For example, due to the central limit theorem,

$$\hat{\theta}_n = n^{-1} \sum_{i=1}^{n} X_i \approx \mathcal{N}(\mu, \sigma^2/n), \;\; X_i \in \mathbb{R},$$

where $\mu = \mathbb{E}[X_i]$, $\sigma^2 = \mathrm{Var}(X_i)$. We then only need to estimate the parameters μ and σ in order to have an approximate distribution for $n^{-1} \sum_{i=1}^{n} X_i$. For the maximum-likelihood estimator in general, estimation of the asymptotic variance is already more subtle. Or for the sample median,

$$\begin{aligned} \hat{\theta}_n &= \mathrm{median}(X_1, \ldots, X_n) \approx \mathcal{N}(\theta, \sigma^2_{asy}/n), \;\; X_i \in \mathbb{R}, \\ \sigma^2_{asy} &= \mathbb{E}[(X-\theta)^2]/(4f^2(\theta)) \end{aligned}$$

where the asymptotic variance already involves quantities like the density f of P at the unknown parameter $\theta = \mathrm{median}(P)$. Estimating this asymptotic variance is already a pretty awkward task, and we should keep in mind that we would then only get the asymptotic answer to the problem of getting the distribution of $\hat{\theta}_n$. Finally, for more complex algorithms, mathematical theory is lacking for obtaining the approximate, asymptotic distribution.

A pioneering step has then be taken by Efron (1979). Suppose we would know what the distribution P is: we could then **simulate** to obtain the distribution of any $\hat{\theta}_n$ with arbitrary accuracy (when simulating enough). Because we do not know the distribution P of the data generating mechanism, we use the **empirical distribution** $\hat{P}_n$ which places probability mass $1/n$ on every data point Z_i, $i = 1, \ldots, n$. The recipe is then to simulate from $\hat{P}_n$: generate simulated data

$$Z_1^*, \ldots Z_n^* \text{ i.i.d. } \sim \hat{P}_n.$$

Such a simulated new data set is called a bootstrap sample. We can now compute our estimator $\hat{\theta}_n^* = g(Z_1^*, \ldots, Z_n^*)$, analogously to (5.1) but based on the bootstrap sample, and we then repeat this many times to get an approximate distribution, e.g. via the histogram of many simulated $\hat{\theta}_n^*$'s.

5.2.1 The bootstrap algorithm

Bootstrapping an estimator as in (5.1) can be done as follows.

1. Generate a bootstrap sample

   $$Z_1^*, \ldots, Z_n^* \text{ i.i.d. } \sim \hat{P}_n.$$

   This can be realized as follows. Do n **uniform random drawings with replacement** from the data set $\{Z_1, \ldots, Z_n\}$, yielding the bootstrap sample.

2. Compute the bootstrapped estimator

   $$\hat{\theta}_n^* = g(Z_1^*, \ldots, Z_n^*),$$

   based on the bootstrap sample; the function $g(\cdot)$ is as in (5.1).

3. Repeat steps 1 and 2 B times to obtain

$$\hat{\theta}_n^{*1}, \ldots, \hat{\theta}_n^{*B}.$$

4. These B bootstrapped estimators in 3 can then be used as approximations for the bootstrap expectation, the bootstrap variance and the bootstrap quantiles:

$$\mathbb{E}^*[\hat{\theta}_n^*] \approx B^{-1} \sum_{i=1}^{B} \hat{\theta}_n^{*i},$$

$$\mathrm{Var}^*(\hat{\theta}_n^*) \approx (B-1)^{-1} \sum_{i=1}^{B} (\hat{\theta}_n^{*i} - B^{-1} \sum_{j=1}^{B} \hat{\theta}_n^{*j})^2,$$

$$\alpha\text{-quantile of distribution of } \hat{\theta}_n^* \approx \text{ empirical } \alpha\text{-quantile of } \hat{\theta}_n^{*1}, \ldots, \hat{\theta}_n^{*B}.$$

The definition of the bootstrap values $\mathbb{E}^*$, Var^* or the bootstrap distribution are discussed next.

5.2.2 The bootstrap distribution

The bootstrap distribution, denoted here by P^*, is the conditional probability distribution which is induced by i.i.d. resampling of the data

$$Z_1^*, \ldots, Z_n^* \text{ i.i.d. } \sim \hat{P}_n, \tag{5.2}$$

given the original data. The fact that we condition on the data allows to treat the bootstrap resampling distribution $\hat{P}_n$, which is the empirical distribution of the data, as a fixed distribution.

Therefore, the bootstrap distribution P^* of $\hat{\theta}_n^* = g(Z_1^*, \ldots, Z_n^*)$ is the distribution which arises when resampling with $\hat{P}_n$ in (5.2) and applying the function g on such a bootstrap sample, exactly as done by simulation in section 5.2.1. (From a theoretical point of view, the bootstrap distribution $\mathcal{P}^*$ can be represented by a multinomial distribution, which contains the information which and how many times of the original data appears again in the bootstrap sample (5.2)).

The bootstrap expectation of $\hat{\theta}_n^*$ is then denoted by $\mathbb{E}^*[\hat{\theta}_n^*]$ which is a conditional expectation given the data. Likewise, $\mathrm{Var}^*(\hat{\theta}_n^*)$ is a conditional variance given the data. Since $\hat{P}_n$ in (5.2) is "close" to the true data-generating probability P, the bootstrap values are "reasonable" estimates for the true quantities. For example, we can use

$$\widehat{\mathrm{Var}(\hat{\theta}_n)} = \mathrm{Var}^*(\hat{\theta}_n^*).$$

This estimate is approximately computed as in step 4 of the bootstrap algorithm in section 5.2.1. Whether such a bootstrap estimate is consistent will be discussed in the following section.

5.2.3 Bootstrap confidence interval: a first approach

The bootstrap is called to be consistent for $\hat{\theta}_n$ if, for all x,

$$\mathbb{P}[a_n(\hat{\theta}_n - \theta) \le x] - \mathbb{P}^*[a_n(\hat{\theta}_n^* - \hat{\theta}_n) \le x] \xrightarrow{P} 0 \ (n \to \infty). \tag{5.3}$$

In classical situations, $a_n = \sqrt{n}$: for example, the maximum-likelihood estimator $\hat{\theta}_n$ satisfies under regularity assumptions

$$\sqrt{n}(\hat{\theta}_{n,MLE} - \theta) \xrightarrow{D} \mathcal{N}(0, I^{-1}(\theta)) \ (n \to \infty),$$

where $I(\theta)$ denotes the Fisher information at θ. Consistency in (5.3) then means

$$\sqrt{n}(\hat{\theta}^*_{n,MLE} - \hat{\theta}_n) \xrightarrow{D^*} \mathcal{N}(0, I^{-1}(\theta)) \text{ in probability } (n \to \infty).$$

> Consistency of the bootstrap typically holds if the limiting distribution of $\hat{\theta}_n$ is Normal, and if the data $Z_1, \ldots, Z_n$ are i.i.d.

Consistency of the bootstrap (usually) implies consistent variance and bias estimation:

$$\frac{\mathrm{Var}^*(\hat{\theta}^*_n)}{\mathrm{Var}(\hat{\theta}_n)} \xrightarrow{P} 1,$$
$$\frac{\mathbb{E}^*[\hat{\theta}^*_n] - \hat{\theta}_n}{\mathbb{E}[\hat{\theta}_n] - \theta} \xrightarrow{P} 1.$$

Moreover, consistent confidence intervals can be construicted.

A two-sided confidence interval with coverage $1 - \alpha$ for a parameter θ is given by

$$[\hat{\theta}_n - q_{1-\alpha/2}, \hat{\theta}_n - q_{\alpha/2}],$$
$$q_\alpha = \alpha\text{-quantile of } \hat{\theta}_n - \theta.$$

This is derived using elementary calculations. In analogy, the bootstrap estimated confidence interval is then defined as

$$[\hat{\theta}_n - \hat{q}_{1-\alpha/2}, \hat{\theta}_n - \hat{q}_{\alpha/2}],$$
$$\hat{q}_\alpha = \alpha\text{-bootstrap quantile of } \hat{\theta}^*_n - \hat{\theta}_n. \tag{5.4}$$

Due to invariance of the quantile:

$$\hat{q}_\alpha = q^*_\alpha - \hat{\theta}_n,$$
$$q^*_\alpha = \alpha\text{-bootstrap quantile of } \hat{\theta}^*_n.$$

Therefore, the bootstrap confidence interval in (5.4) becomes

$$[2\hat{\theta}_n - q^*_{1-\alpha/2}, 2\hat{\theta}_n - q^*_{\alpha/2}].$$

This is **not the same as simply taking the quantiles of the bootstrap distribution**, i.e. the simple-minded $[q^*_{\alpha/2}, q^*_{1-\alpha/2}]$) is "backwards" and often less appropriate. The derivation which we gave for this "unintuitive" fact hinges on the consistency of the bootstrap in (5.3).

Better bootstrap confidence intervals tan (5.4) exist which often have better coverage accuracy; at the price of being somewhat more difficult to implement. See also section 5.3.

method	intercept	LOGVISC	ASPH	BASE	RUN	FINES	VOIDS
classical	[-10.86,-0.71]	[-0.66,-0.36]	[0.60,1.70]	[-0.44,0.91]	[-1.23,-0.01]	[-0.02,0.02]	[0.09,0.54]
bootstrap	[**-11.97,0.99**]	[-0.67,-0.36]	[0.41,1.78]	[-0.37,0.87]	[**-1.26,0.09**]	[-0.01,0.02]	[0.10,0.60]

Table 5.1: Classical 95% confidence intervals and 95% bootstrap confidence intervals (using $B = 1000$) from (5.4) for the 7 coefficients in a linear model for the asphalt dataset from section 1.5.1. In bold are the bootstrap confidence intervals which suggest non-significance while the classical confidence intervals (or t-tests) would suggest significance.

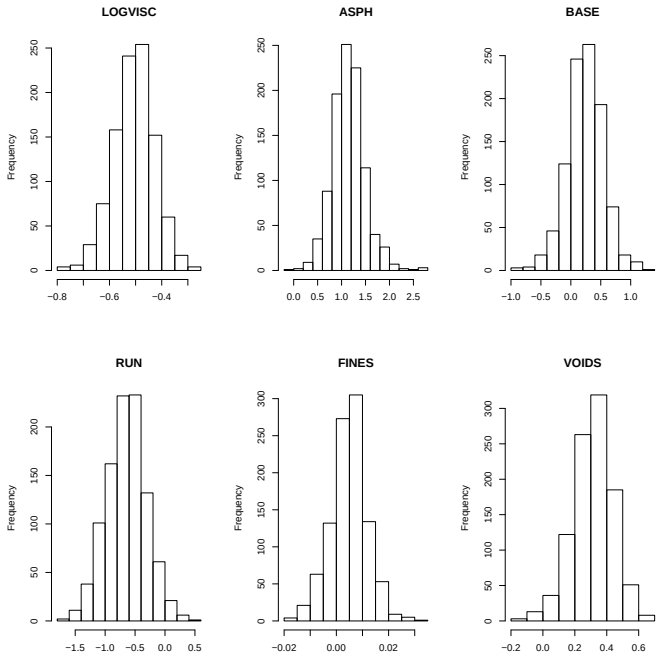


Figure 5.1: Histograms of $B = 1000$ bootstrap values $\hat{\beta}_j^*$ for the 6 coefficients in the linear model about asphalt quality from section 1.5.1.

An example

We consider here the example about asphalt quality from section 1.5.1, where a linear model has been used to model the log of rutting as a function of 6 predictor variables.

The standard confidence intervals based on normally distributed data (or on asymptotic considerations) are (including an intercept)

$$\hat{\beta}_j \pm t_{n-p-1;1-\alpha/2}\widehat{s.e.}(\hat{\beta}_j),$$

where $t_{n-p-1;\alpha}$ denotes the α-quantile of a t_{n-p-1} distribution. In this example, $n = 31$, $p = 6$ when using $\alpha = 0.05$, we get $t_{n-7;1-\alpha/2} = 2.063899$. The confidence intervals are then given in Table 5.2.3. The bootstrap distributions for the 6 coefficients corresponding to the 6 predictor variables are exhibited in Figure 5.1.

5.2.4 Bootstrap estimate of the generalization error

Consider the problem of estimating the generalization error

$$\mathbb{E}[\rho(Y_{new}, \hat{m}(X_{new}))]$$

in (4.1), where ρ is a loss function such as $\rho(y,m) = |y-m|^2$ and $\hat{m}(\cdot)$ could be a regression estimator.

The bootstrap approach to estimate the generalization error is from a conceptual view as follows.

1. Generate

$$(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*), (X_{new}^*, Y_{new}^*) \text{ i.i.d. } \sim \hat{P}_n,$$

where $\hat{P}_n$ is the empirical distribution of the original data $(X_1, Y_1), \ldots, (X_n, Y_n)$.

2. Compute the bootstrapped estimator $\hat{m}^*(\cdot)$ based on $(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$.

3. Compute the bootstrap generalization error

$$\mathbb{E}^*[\rho(Y_{new}^*, \hat{m}^*(X_{new}^*))],$$

where $\mathbb{E}^*$ is with respect to all the bootstrap variables $\text{train}^* = (X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$ **and** $\text{test}^* = (X_{new}^*, Y_{new}^*)$. Use this as an estimate of the true generalization error.

The bootstrap generalization error can be re-written as follows:

$$\begin{aligned} & \mathbb{E}^*[\rho(Y_{new}^*, \hat{m}(X_{new}^*))] = \mathbb{E}^*_{\text{train}^*}\left[\mathbb{E}^*_{\text{test}^*}[\rho(Y_{new}^*, \hat{m}(X_{new}^*))|\text{train}^*]\right] \\ = \; & \mathbb{E}^*_{\text{train}^*}[n^{-1}\sum_{i=1}^n \rho(Y_i, \hat{m}^*(X_i))] = n^{-1}\sum_{i=1}^n \mathbb{E}^*[\rho(Y_i, \hat{m}^*(X_i))]. \end{aligned} \tag{5.5}$$

The first equality on the second line follows because: (i) $\text{test}^* = (X_{new}^*, Y_{new}^*)$ is independent (with respect to the bootstrap distribution) from train^*, and hence the inner conditional expectation is a non-conditional expectation using $\hat{m}^*(\cdot)$ as fixed (non-random because we condition on the bootstrap training data train^*); (ii) the bootstrap expectation $\mathbb{E}^*[g(X^*, Y^*)] = n^{-1}\sum_{i=1}^n g(X_i, Y_i)$ for any function $g(\cdot)$.

Therefore, the bootstrap generalization error as represented in (5.5) is the average of the bootstrap errors over the observed original data (X_i, Y_i). In particular, there is no need to generate (X_{new}^*, Y_{new}^*) as conceptually introduced above in step 1. The practical algorithm then looks as follows.

1. Generate $(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$ by resampling with replacement from the original data.

2. Compute the bootstrapped estimator $\hat{m}^*(\cdot)$ based on $(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$.

3. Evaluate $\text{err}^* = n^{-1}\sum_{i=1}^n \rho(Y_i, \hat{m}^*(X_i))$.

4. Repeat steps 1–3 B times to obtain $\text{err}^{*1}, \ldots, \text{err}^{*B}$. Approximate the bootstrap generalization error in (5.5) by

$$B^{-1}\sum_{i=1}^B \text{err}^{*i},$$

and use it as an estimate for the true generalization error in (4.1).

5.3 Double bootstrap

We describe here how the bootstrap can be used twice (or multiple times) aiming to improve a bootstrap confidence interval. The same idea can also be used for potentially improving other bootstrap estimates than confidence intervals.

Suppose we wish to construct a $(1-\alpha)$-confidence interval for a parameter θ based on some estimator $\hat{\theta}$. The bootstrap interval $I^*(1-\alpha)$, as defined in (5.4), is not exact, i.e.

$$\mathbb{P}[\theta \in I^*(1-\alpha)] = 1 - \alpha + \Delta_n,$$

with some approximation error Δ_n which will diminish as $n \to \infty$.

The main idea is now as follows: when changing the nominal coverage to $1-\alpha'$ and using $I^*(1-\alpha')$, we can get an exact actual coverage

$$\mathbb{P}[\theta \in I^*(1-\alpha')] = 1 - \alpha.$$

The problem is that α' is unknown. But another level of bootstrap can be used to **estimate** α', denoted by $\widehat{\alpha'}$, which typically achieves

$$\mathbb{P}[\theta \in I^*(\widehat{\alpha'})] = 1 - \alpha + \Delta_n',$$

where Δ_n' is typically smaller than the approximation error Δ_n above.

A second level of bootstrap

In case where the original data $Z_1, \ldots, Z_n$ is replaced by n i.i.d. bootstrap realizations $Z_1^*, \ldots, Z_n^*$, we can get an exact answer for the level α' above. We can proceed in analogy to the setting above, step by step.

First, suppose the data is $Z_1^*, \ldots, Z_n^*$. By using the bootstrap for the bootstrap data $Z_1^*, \ldots, Z_n^*$, i.e. a second level bootstrap with $Z_1^{**}, \ldots, Z_n^{**}$ (see below), we can construct a confidence interval $I^{**}(1-\alpha)$ as in (5.4). We can now inspect the actual coverage for this second level bootstrap interval:

$$\mathbb{P}^*[\hat{\theta}_n \in I^{**}(1-\alpha)] = h^*(1-\alpha)$$

for some function $h^* : [0,1] \to [0,1]$ which is increasing. Now use

$$1 - \alpha'^* = h^{*-1}(1-\alpha) \tag{5.6}$$

so that

$$\mathbb{P}^*[\hat{\theta}_n \in I^{**}(1-\alpha'^*)] = h^*(h^{*-1}(1-\alpha)) = 1-\alpha$$

is an exact confidence interval "in the bootstrap world" for the "parameter" $\hat{\theta}_n$. Therefore, the bootstrap estimate for the adjusted nominal coverage level is

$$\widehat{1-\alpha'} = 1 - \alpha'^* \text{ from (5.6)}.$$

Computation of bootstrap adjusted nominal level $1-\alpha'^*$

We can use a double (two-level) bootstrap scheme to approximately compute the value $1-\alpha'^*$ in (5.6).

1. Draw a bootstrap sample $Z_1^*, \ldots, Z_n^*$ by resampling with replacement.

   (a) Compute a bootstrap interval. That is, generate a second level bootstrap sample

   $$Z_1^{**}, \ldots, Z_n^{**}$$

   by resampling with replacement from $Z_1^*, \ldots, Z_n^*$. Then, analogous to (5.4), construct the double bootstrap confidence interval

   $$I^{**}(1-\alpha) = [\hat{\theta}^* - \hat{q}^*_{1-\alpha/2}, \hat{\theta}^* - \hat{q}^*_{\alpha/2}],$$

   where $\hat{q}^*_\alpha = \alpha-$ quantile of $\hat{\theta}^{**} - \hat{\theta}^*$, $\hat{\theta}^{**} = g(Z_1^{**}, \ldots, Z_n^{**})$. This is computed by repeating step 1(a) B times to approximate $\hat{q}^*_\alpha$ by the empirical α-quantile of $\hat{\theta}^{**1} - \hat{\theta}^*, \ldots, \hat{\theta}^{**B} - \hat{\theta}^*$.

   (b) Evaluate whether the "parameter" $\hat{\theta}$ in the "bootstrap world" is in $I^{**}(1-\alpha)$: i.e., consider

   $$\text{cover}^*(1-\alpha) = \mathbf{1}_{[\hat{\theta} \in I^{**}(1-\alpha)]}.$$

2. Repeat steps 1(a)-(b) M times to obtain $\text{cover}^{*1}(1-\alpha), \ldots, \text{cover}^{*M}(1-\alpha)$. This amounts to M times first level bootstrap replications $\mathbf{Z}^{*m} = Z_1^{*m}, \ldots, Z_n^{*m}$ $(m = 1, \ldots, M)$ and for **each** $\mathbf{Z}^{*m}$, we run B second level bootstrap replications for step 1(a). Use

   $$M^{-1} \sum_{i=1}^{M} \text{cover}^{*i}(1-\alpha)$$

   as an approximation for $\mathbb{P}^*[\hat{\theta} \in I^{**}(1-\alpha)]$.

3. Vary α to find α'^* such that

   $$M^{-1} \sum_{i=1}^{M} \text{cover}^{*i}(1-\alpha'^*) = 1-\alpha.$$

   The search for α'^* can be done on a grid.

The total amount of computation requires $B \cdot M$ bootstrap samples. In case where the bootstrap interval in (5.4) is computed with B bootstrap samples, and hence also the interval I^{**} in step 1(a), the adjustment with the double bootstrap may be less important and it is then reasonable to use $M < B$ since the magnitude of M only determines the approximation for computing the actual level $\mathbb{P}^*[\hat{\theta} \in I^{**}(1-\alpha)]$ (for I^{**} computed with B bootstrap replications).

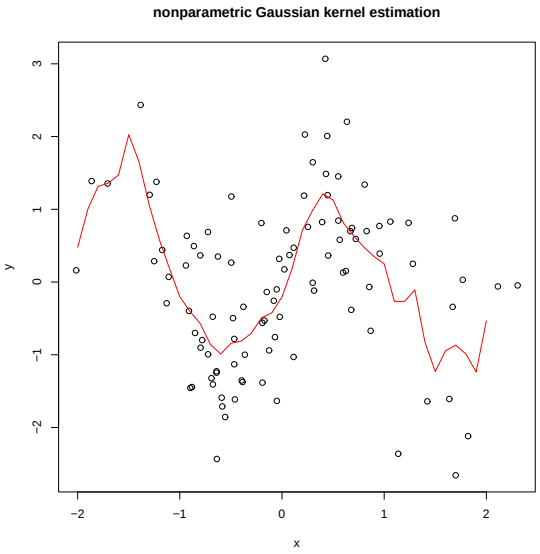


Figure 5.2: Data ($n = 100$) and estimated curve (red) using a Nadaraya Watson Gaussian kernel estimator with bandwidth $h = 0.25$.

An example

We illustrate now the double bootstrap for confidence intervals in curve estimation. Figure 5.2 displays the data, having sample size $n = 100$, and a curve estimator.

Figure 5.3 then shows how the double bootstrap is used to estimate the actual coverage: displayed is an approximation of $\mathbb{P}^*[\hat{\theta}_n \in I^{**}(1-\alpha)]$ for various nominal levels $1-\alpha$. It also indicates the values for the corrected levels $1-\alpha'^*$ and it also demonstrates the effect when using a double-bootstrap corrected confidence interval instead of an ordinary interval.

5.4 Model-based bootstrap

Efron's nonparametric bootstrap can be viewed as simulating from the empirical distribution $\hat{P}_n$: that is, we simulate from a very general estimated nonparametric model, where the model says that the data is i.i.d. distributed with an unknown distribution P.

5.4.1 Parametric bootstrap

Instead of such a general nonparametric model, we sometimes assume that the data are realizations from

$$Z_1, \ldots, Z_n \text{ i.i.d. } \sim P_\theta,$$

where P_θ is given up to an unknown parameter (vector) θ.

As one among very many examples: the data could be real-valued assumed to be from the parametric model

$$X_1, \ldots, X_n \text{ i.i.d. } \sim \mathcal{N}(\mu, \sigma^2), \ \theta = (\mu, \sigma^2).$$

In order to simulate from the parametric model, we first estimate the unknown parameter θ by $\hat{\theta}$ such as least squares in regression or maximum likelihood in general. The

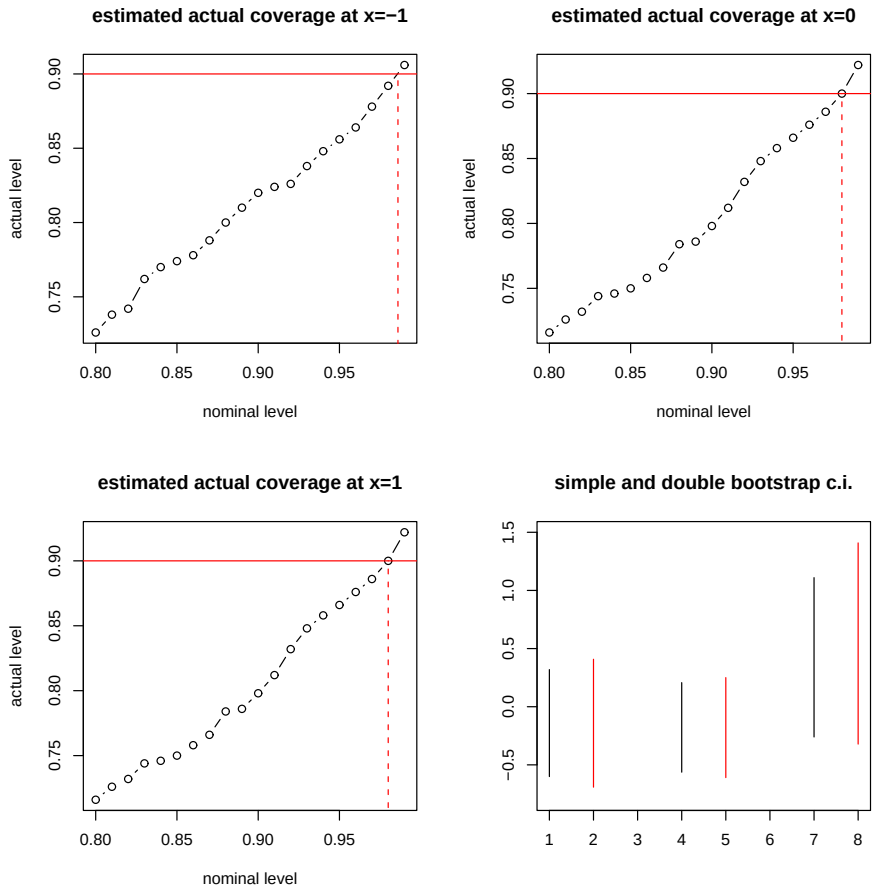


Figure 5.3: Double bootstrap confidence intervals for nonparametric curve at three predictor points $x \in \{-1, 0, 1\}$. The data ($n = 100$) and estimated curve are shown in Figure 5.2. The first three panels show the estimated actual coverage of a bootstrap confidence interval by using the double bootstrap. The values $1 - \alpha'^*$ (for nominal level $1 - \alpha = 0.9$) are 0.986, 0.98, 0.98 for the points $x = -1, 0, 1$, respectively. The fourth panel shows the ordinary bootstrap confidence intervals (solid line) and the double bootstrap corrected versions (dotted line, in red) for $x = -1$ (left), $x = 0$ (middle) and $x = 1$ (right). The double bootstrap was used with $B = 500$ and $M = 100$.

parametric bootstrap then proceeds by using

$$Z_1^*, \ldots, Z_n^* \text{ i.i.d.} \sim P_{\hat{\theta}},$$

instead of (5.2). Everything else, e.g. construction of confidence intervals, can then be done exactly as for Efron's nonparametric bootstrap.

Advantages and disadvantages

Why should we choose the parametric instead of the nonparametric bootstrap? The answer is "classical": if the parametric model is a very good description for the data, then the parametric bootstrap should yield more accurate variance estimates or confidence intervals since $P_{\hat{\theta}}$ is then "closer" to the true data-generating P than the nonparametric empirical distribution $\hat{P}_n$. Particularly when sample size n is small, the nonparametric estimate $\hat{P}_n$ may be poor. On the other hand, the nonparametric bootstrap is not (or less) sensitive to model-misspecification.

5.4.2 Model structures beyond i.i.d. and the parametric bootstrap

Linear model with fixed predictors

For example, a linear model with fixed predictors $x_i \in \mathbb{R}^p$ and Gaussian errors

$$\begin{aligned} &Y_i = \beta^T x_i + \varepsilon_i \ (i = 1, \ldots, n), \\ &\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d } \sim \mathcal{N}(0, \sigma^2), \ \theta = (\beta, \sigma^2) \end{aligned}$$

is a parametric model. The bootstrap sample can then be constructed as follows:

1. Simulate $\varepsilon_1^*, \ldots, \varepsilon_n^*$ i.i.d $\sim \mathcal{N}(0, \hat{\sigma}^2)$.
2. Construct

$$Y_i^* = \hat{\beta}^T x_i + \varepsilon_i^*, \ i = 1, \ldots, n.$$

   The parametric bootstrap regression sample is then

$$(x_1, Y_1^*), \ldots, (x_n, Y_n^*),$$

   where the predictors x_i are as for the original data.

Autoregressive models for time series

A Gaussian autoregressive model of order p for stationary time series is

$$\begin{aligned} &X_t = \sum_{j=1}^{p} \phi_j X_{t-j} + \varepsilon_t \ (t = 1, \ldots, n), \\ &\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d } \sim \mathcal{N}(0, \sigma^2), \end{aligned}$$

where $X_t \in \mathbb{R}$. Such a model produces correlated observations and is widely used for describing time-dependent observations. Parametric bootstrapping can then be done as follows:

1. Generate $\varepsilon_1^*, \ldots, \varepsilon_{n+m}^*$ i.i.d. $\sim \mathcal{N}(0, \hat{\sigma}^2)$ with $m \approx 1000$.

2. Construct recursively, starting with $X_0^* = X_{-1}^* = \ldots = X_{-p+1}^* = 0$,

$$X_t^* = \sum_{j=1}^{p} \hat{\phi}_j X_{t-j}^* + \varepsilon_t^*, \ t = 1, \ldots, n+m.$$

3. Use the bootstrap sample

$$X_{m+1}^*, \ldots, X_{n+m}^*.$$

The reason to throw away the first values $X_1^*, \ldots X_m^*$ is to obtain a bootstrap sample which is approximately a stationary process (by choosing m large, the arbitrary starting values in step 2 will be almost forgotten).

5.4.3 The model-based bootstrap for regression

A compromise between Efron's non- and the parametric bootstrap for regression is given by assuming possibly non-Gaussian errors. The model for the original data is

$$Y_i = m(x_i) + \varepsilon_i,$$
$$\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d} \sim P_\varepsilon,$$

where P_ε is unknown with expectation 0. The regression function $m(\cdot)$ may be parametric or nonparametric. The model-based bootstrap works then as follows:

1. Estimate $\hat{m}$ from the original data and compute the residuals $r_i = Y_i - \hat{m}(x_i)$.

2. Consider the centered residuals $\tilde{r}_i = r_i - n^{-1} \sum_{i=1}^n r_i$. In case of linear regression with an intercept, the residuals are already centered. Denote the empirical distribution of the centered residuals by $\hat{P}_{\tilde{r}}$.

3. Generate

$$\varepsilon_1^*, \ldots, \varepsilon_n^* \text{ i.i.d.} \sim \hat{P}_{\tilde{r}}.$$

Note that $\hat{P}_{\tilde{r}}$ is an estimate of P_ε.

4. Construct the bootstrap response variables

$$Y_i^* = \hat{m}(x_i) + \varepsilon_i^*, \ i = 1, \ldots, n,$$

and the bootstrap sample is then $(x_1, Y_1^*), \ldots, (x_n, Y_n^*)$.

Having the bootstrap sample from step 4, we can then proceed as for Efron's nonparametric bootstrap for constructing variance estimates or confidence intervals.

The advantage of the model-based bootstrap is that we do not rely on a Gaussian error assumption. The same discussion then applies about advantages and disadvantages as in section 5.4.1.

Chapter 6

Classification

6.1 Introduction

Often encountered in applications is the situation where the response variable Y takes values in a finite set of labels. For example, the response Y could encode the information whether a patient has disease type A, B or C; or it could describe whether a customer responds positively about a marketing campaign.

We always encode such information about classes or labels by the numbers $0, 1, \ldots, J-1$. Thus, $Y \in \{0, 1, \ldots, J-1\}$, without any ordering among these numbers $0, 1, \ldots, J-1$.

Given data which are realizations from

$$(X_1, Y_1), \ldots, (X_n, Y_n) \text{ i.i.d.},$$

the goal is often to assign the probabilities $\pi_j(x) = \mathbb{P}[Y = j | X = x]$ $(j = 0, 1, \ldots, J-1)$, which is similar to the regression function $m(x) = \mathbb{E}[Y|X = x]$ in regression. The multivariate function $\pi_j(\cdot)$ then also allows to predict the class Y_{new} at a new observed predictor X_{new}.

6.2 The Bayes classifier

A classifier $\mathcal{C} : \mathbb{R}^p \to \{0, 1, \ldots, J-1\}$ is a function which assigns to a predictor $X \in \mathbb{R}^p$ a class or label which is a prediction for the corresponding Y.

The quality of a classifier is often measured by the expected zero-one test set error:

$$\mathbb{P}[\mathcal{C}(X_{new}) \neq Y_{new}].$$

In case where $\mathcal{C} = \widehat{\mathcal{C}}$ is estimated from training data, we consider the generalization error

$$\mathbb{P}_{\text{train},(X_{new},Y_{new})}[\widehat{\mathcal{C}}(X_{new}) \neq Y_{new}].$$

The optimal classifier with respect to the zero-one error is the **Bayes classifier**

$$\mathcal{C}_{Bayes}(x) = \underset{0 \leq j \leq J-1}{\arg\max}\, \pi_j(x). \tag{6.1}$$

Its corresponding expected zero-one test set error is called the **Bayes risk**

$$\mathbb{P}[\mathcal{C}_{Bayes}(X_{new}) \neq Y_{new}].$$

In practice, we do not know $\pi_j(\cdot)$. Various methods and models are then used to come up with multivariate function estimates, either parametric or nonparametric, to obtain $\hat{\pi}_j(\cdot)$. With this, we can then estimate a classifier by plugging into the Bayes classifier

$$\widehat{\mathcal{C}}(x) = \operatorname*{arg\,max}_{0 \le j \le J-1} \hat{\pi}_j(x). \tag{6.2}$$

Such estimated classifiers are widely used, by using various models for $\pi_j(\cdot)$. However, there are also direct ways to come up with estimated classifiers without trying to estimate the conditional probability function $\pi_j(\cdot)$: an important class of examples are the support vector machines (which we will not discuss in this course).

6.3 The view of discriminant analysis

6.3.1 Linear discriminant analysis

For the so-called linear discriminant analysis, we assume the following model:

$$\begin{aligned} &X|Y = j \sim \mathcal{N}_p(\mu_j, \Sigma), \\ &\mathbb{P}[Y = j] = p_j, \ \sum_{j=0}^{J-1} p_j = 1, \ j = 0, 1, \ldots, J-1. \end{aligned} \tag{6.3}$$

The conditional distribution of $Y|X$ can then be computed by the Bayes formula

$$\mathbb{P}[Y = j|X = x] = \pi_j(x) = \frac{f_{X|Y=j}(x) p_j}{\sum_{k=0}^{J-1} f_{X|Y=k}(x) p_k}, \tag{6.4}$$

where $f_{X|Y=j}(\cdot)$ denotes the density of the p-dimensional Gaussian distribution $\mathcal{N}_p(\mu_j, \Sigma)$. We can interpret this conditional distribution as the a-posteriori distribution of Y given X by using the a-priori distribution p_j for Y.

The unknown parameters in (6.4) are μ_j and Σ which can be estimated by standard moment estimators:

$$\begin{aligned} \hat{\mu}_j &= \sum_{i=1}^{n} X_i \mathbf{1}_{[Y_i=j]} / \sum_{i=1}^{n} \mathbf{1}_{[Y_i=j]}, \\ \hat{\Sigma} &= \sum_{j=0}^{J-1} \sum_{i=1}^{n} (X_i - \hat{\mu}_j)(X_i - \hat{\mu}_j)^T \mathbf{1}_{[Y_i=j]} / (n - J). \end{aligned}$$

Moreover, we need to specify the (a-priori) distribution for Y: quite often, one takes $\hat{p}_j = n^{-1} \sum_{i=1}^{n} \mathbf{1}_{[Y_i=j]}$. Using these parameter estimates, we obtain a classifier via formula (6.4) and (6.2):

$$\begin{aligned} \widehat{\mathcal{C}}_{lin.discr.}(x) &= \operatorname*{arg\,max}_{0 \le j \le J-1} \hat{\delta}_j(x), \\ \hat{\delta}_j(x) &= x^T \hat{\Sigma}^{-1} \hat{\mu}_j - \hat{\mu}_j^T \hat{\Sigma}^{-1} \hat{\mu}_j / 2 + \log(\hat{p}_j). \end{aligned}$$

This classifier is called "linear discriminant classifier" because the estimated decision functions $\hat{\delta}_j(\cdot)$ are **linear** in the predictor variables $\mathbf{x}$.

6.3.2 Quadratic discriminant analysis

The model underlying linear discriminant analysis assumes equal covariances for all the groups, see formula (6.3). More generally, we sometimes assume

$$X|Y = j \sim \mathcal{N}_p(\mu_j, \Sigma_j),$$
$$\mathbb{P}[Y = j] = p_j, \ \sum_{j=0}^{J-1} p_j = 1, \ j = 0, 1, \ldots, J-1,$$

with non-equal covariances for all the groups $j \in \{0, 1, \ldots, J-1\}$. Analogously to linear discriminant analysis, we then obtain discriminant functions which are **quadratic** in the predictor variables x,

$$\hat{\delta}_j(x) = -\log(\det(\hat{\Sigma}_j))/2 - (x - \hat{\mu}_j)^T \hat{\Sigma}_j^{-1}(x - \hat{\mu}_j)/2 + \log(\hat{p}_j).$$

Such quadratic discriminant classifiers are more flexible and general than their linear "cousins": but the price to pay for this flexibility are $Jp(p+1)/2$ parameters for all covariance matrices Σ_j $(j = 0, 1, \ldots, J-1)$ instead of $p(p+1)/2$ for one Σ in linear discriminant analysis. Particularly when p is large, quadratic discriminant analysis typically overfits (too large variability).

6.4 The view of logistic regression

As we have seen in (6.1), all we need to know for a good classifier is a good estimator for the conditional probabilities $\pi_j(\cdot)$.

6.4.1 Binary classification

For the case with binary response $Y \in \{0, 1\}$, the conditional probability function

$$\pi(x) = \mathbb{P}[Y = 1|X = x]$$

provides the full information about the conditional distribution of Y given X (since $\mathbb{P}[Y = 0|X = x] = 1 - \pi(x)$).

The logistic model for $\pi(\cdot)$ in general is

$$\log\left(\frac{\pi(x)}{1 - \pi(x)}\right) = g(x),$$
$$g : \mathbb{R}^p \to \mathbb{R}. \tag{6.5}$$

Note that the so-called logistic transform $\pi \mapsto \log(\pi/(1-\pi))$ maps the interval $(0, 1)$ to the real line $\mathbb{R}$. Models for real-valued functions can thus be used for $g(\cdot)$.

Linear logistic regression

In analogy to linear regression in chapter 1, a popular and simple model for $g(\cdot)$ is

$$g(x) = \sum_{j=1}^{p} \beta_j x_j. \tag{6.6}$$

The model in (6.5) with $g(\cdot)$ from (6.6) is called linear logistic regression.

Fitting of the parameters is usually done by maximum-likelihood. For fixed predictors x_i, the probability structure of the response variables is

$$Y_1, \ldots, Y_n \text{ independent}, \; Y_i \sim \text{ Bernoulli}(\pi(x_i)).$$

The likelihood is thus

$$L(\beta; (x_1, Y_1), \ldots, (x_n, Y_n)) = \prod_{i=1}^{n} \pi(x_i)^{Y_i} (1 - \pi(x_i))^{1-Y_i},$$

and the negative log-likelihood becomes

$$\begin{aligned} -\ell(\beta; (x_1, Y_1), \ldots, (x_n, Y_n)) &= -\sum_{i=1}^{n} (Y_i \log(\pi(x_i)) + (1 - Y_i) \log(1 - \pi(x_i))) \\ &= -\sum_{i=1}^{n} \left(Y_i \sum_{j=1}^{p} \beta_j x_{ij} - \log(\exp(\sum_{j=1}^{p} \beta_j x_{ij}) + 1) \right). \end{aligned}$$

Minimization of the negative log-likelihood is a **nonlinear** problem. It is usually solved numerically by versions of Newton's gradient descent method which then yield the maximal likelihood estimate $\hat{\beta}_{p \times 1}$.

Asymptotic inference

Based on classical theory for maximum likelihood estimation, the distribution of the estimated coefficients $\hat{\beta}$ can be derived from an asymptotic point of view where the sample size $n \to \infty$. The output in `R` then yields a summary of the estimated coefficients, of their estimated standard errors $\widehat{s.e.}(\hat{\beta}_j)$, of the individual t-test statistics $\hat{\beta}_j / \widehat{s.e.}(\hat{\beta}_j)$ (which are asymptotically $\mathcal{N}(0,1)$ distributed under the null-hypothesis $H_{0,j} : \beta_j = 0$) and the P-values for the individual t-tests for the null-hypotheses $H_{0,j} = 0$ $(j = 1, \ldots, p)$.

As an example, we consider a dataset about survival of prenatal babies (represented by the response variable $Y \in \{0, 1\}$) as a function of age (in weeks), weight (in grams) and 3 other clinical variables.

Fitting a linear logistic regression model can be done in `R` using the function `glm`:

```
baby.dat <- read.table("http://stat.ethz.ch/Teaching/Datasets/baby.dat",header=T)

fit <- glm(Survival~.,data=baby.dat,family="binomial")

summary(fit)
Call:
glm(formula = Survival ~ ., family = "binomial", data = baby.dat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3992  -0.7393   0.4221   0.7833   1.9444

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.0911829 14.2507067  -0.217   0.8283
Weight       0.0037335  0.0008415   4.437 9.13e-06 ***
```

```
Age          0.1587773  0.0757801   2.095   0.0361 *
X1.Apgar     0.1159401  0.1103167   1.051   0.2933
X5.Apgar     0.0611633  0.1197444   0.511   0.6095
pH          -0.7381402  1.8894030  -0.391   0.6960
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 319.28  on 246  degrees of freedom
Residual deviance: 236.14  on 241  degrees of freedom
AIC: 248.14

Number of Fisher Scoring iterations: 3
```

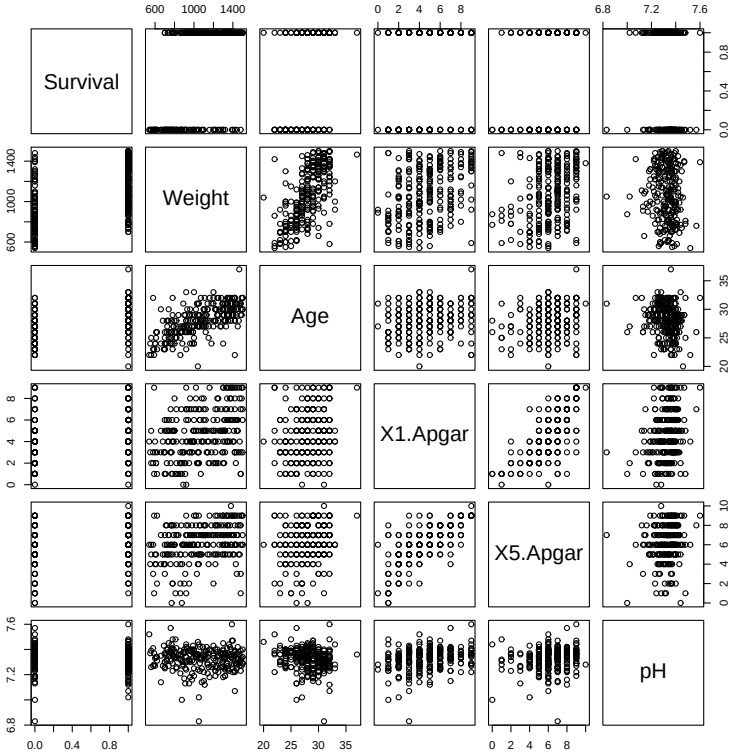


Figure 6.1: Datasets about survival of 247 prenatal babies as a function of age (in weeks), weight (in grams) and 3 other clinical variables.

As seen from the P-values for the individual hypotheses $H_{0,j} : \beta_j = 0$, the predictor variables weight and age, which are strongly correlated, turn out to be significant for describing whether a prenatal baby will survive.

For classification, we can extract the probabilities with the function `predict`:

```
predict(fit,type="response")
```

which yields the estimated probabilities $\hat{\pi}(x_i),\ i = 1, \ldots, n$. Thus, the average in-sample classification $n^{-1} \sum_{i=1}^{n} \mathbf{1}_{[Y_i = \widehat{\mathcal{C}}(x_i)]}$ accuracy is given by

```
mean(predict(fit,type="response")>0.5 == Survival)
```

which turns out to be 0.789.

Linear logistic regression or LDA?

In linear logistic regression, the model for the log-odds (the logit-transform)

$$\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \sum_{j=1}^{p} \beta_j x_j$$

is *linear* in the predictors.But also the log-odds from LDA in model (6.4) yield a linear model in the predictor variables which is a consequence of the Gaussian assumption. Thus, the two methods are quite similar.

The linear logistic regression does not make any assumptions on the predictor variables such a multivariate Gaussianity; but the restriction comes in by requiring a linear log-odds function. Logistic regression can also be used when having factor variables (e.g. $x \in \{0, 1, \ldots, K\}$) as predictors.

While LDA does make a Gaussian assumption for the predictors, it can also be used as a "linear technique" in case of non-Gaussian predictors (even with factors). Empirically, LDA and linear logistic regression yield similar answers, even for non-Gaussian predictors, with respect to classification accuracy.

6.4.2 Multiclass case

Logistic regression cannot be directly applied to the multiclass case with $Y \in \{0, 1, \ldots J-1\}$ and $J > 2$. But we can always encode a multiclass problem with J classes as J binary class problems by using

$$Y_i^{(j)} = \begin{cases} 1 & \text{if } Y_i = j, \\ 0 & \text{otherwise.} \end{cases}$$

This means that we consider class j against all remaining classes.

We can then run logistic regressions

$$\log\left(\frac{\pi_j(x)}{1-\pi_j(x)}\right) = \sum_{r=1}^{p} \beta_r^{(j)} x_r$$

yielding estimates

$$\hat{\pi}_j(x) = \frac{\exp(\sum_{r=1}^{p} \hat{\beta}_r^{(j)})}{1 + \exp(\sum_{r=1}^{p} \hat{\beta}_r^{(j)})}. \tag{6.7}$$

The estimates $\hat{\pi}_j(\cdot)$ will not sum up to one: but a normalization will do the job,

$$\tilde{\pi}_j(x) = \frac{\hat{\pi}_j(x)}{\sum_{j=0}^{J-1} \hat{\pi}_j(x)}.$$

Other ways are also possible to reduce a J class problem into several binary class problems. Instead of modelling class j against the rest, we can also model class j against another class k for all pairs (j, k) with $j \neq k$. This will require fitting logistic models $\binom{J}{2}$ times (involving $\binom{J}{2} \cdot p$ estimated parameters), instead of J times in the one against the rest approach. We should keep in mind that the models are different: in the one against the rest approach, the coefficients in (6.7) describe the effect of the predictors for distinguishing class j from all others, and in a pairwise approach, we would model the distinction between two classes.

Chapter 7

Flexible regression and classification methods

7.1 Introduction

The curse of dimensionality makes it virtually impossible to estimate a regression function $m(x) = \mathbb{E}[Y|X = x]$ or the probability functions $\pi_j(x) = \mathbb{P}[Y = j|X = x]$ (for classification) in a full nonparametric way without making some structural assumptions.

We are going to describe some flexible models and methods which are of nonparametric nature but making some structural assumptions. In the sequel, we will denote either a regression function $\mathbb{E}[Y|X = x]$ or the logit transform in a binary classification problem $\log(\pi(x)/(1 - \pi(x)))$ by

$$g(\cdot) : \mathbb{R}^p \to \mathbb{R}.$$

7.2 Additive models

Additive models require additivity of the function: the model is

$$g_{add}(x) = \mu + \sum_{j=1}^{p} g_j(x_j), \ \mu \in \mathbb{R}$$
$$g_j(\cdot) : \mathbb{R} \to \mathbb{R}, \ \mathbb{E}[g_j(X_j)] = 0 \ (j = 1, \ldots, p).$$

The functions $g_j(\cdot)$ are fully nonparametric. The requirement about $\mathbb{E}[g_j(X_j)] = 0$ yields an identifiable model; note that otherwise we could add and subtract constants into the one-dimensional functions $g_j(\cdot)$.

Additive models are generalizations of linear models. While the additive functions are very general, they do not allow for interaction terms such as $g_{j,k}(x_j, x_k)$. The curse of dimensionality is *not* present in additive models. When assuming continuous second derivatives for all the functions g_j, it can be shown that some estimators $\hat{g}_{add}(\cdot)$ have mean square error rate $O(n^{-4/5})$ as for estimating a single one-dimensional function from $\mathbb{R}$ to $\mathbb{R}$.

7.2.1 Backfitting for additive regression models

Backfitting is a very general tool for estimation in additive (and other) structures. We can use "any" nonparametric smoothing technique for estimating one-dimensional functions.

A smoother against the predictor variables $X_{1j}, \ldots X_{nj}$ is denoted by the hat operator

$$\mathcal{S}_j : (U_1, \ldots, U_n)^T \mapsto (\hat{U}_1, \ldots, \hat{U}_n)^T \ (j = 1, \ldots p)$$

for any response vector $(U_1, \ldots, U_n)^T$. The subscript j indicates that smoothing is done against the jth predictor variable. For example, $\mathcal{S}_j$ could be smoothing splines with the same degrees of freedom for all j, e.g. equal to 5 or estimated by (generalized) cross-validation. Or they could be Nadaraya-Watson Gaussian kernel estimators with the same bandwidth which may be estimated by (generalized) cross-validation.

Backfitting for additive regression then works as follows.

1. Use $\hat{\mu} = n^{-1} \sum_{i=1}^n Y_i$. Start with $\hat{g}_j(\cdot) \equiv 0$ for all $j = 1, \ldots, p$.

2. Cycle through the indices $j = 1, 2, \ldots, p, 1, 2, \ldots, p, 1, 2, \ldots$ while computing

$$\hat{\mathbf{g}}_j = \mathcal{S}_j(\mathbf{Y} - \hat{\mu} - \sum_{k \neq j} \hat{\mathbf{g}}_k),$$

   where $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ and $\hat{\mathbf{g}}_j = (\hat{g}_j(X_{1j}), \ldots, \hat{g}_j(X_{nj}))^T$. Stop the iterations if the individual functions $\hat{g}_j(\cdot)$ do not change much anymore, e.g.

$$\frac{\|\hat{\mathbf{g}}_{j,new} - \hat{\mathbf{g}}_{j,old}\|_2}{\|\hat{\mathbf{g}}_{j,old}\|_2} \leq \text{tol}$$

   where tol is a tolerance such as 10^{-6}.

3. Normalize the functions

$$\tilde{g}_j(\cdot) = \hat{g}_j(\cdot) - n^{-1} \sum_{i=1}^n \hat{g}_j(X_{ij}.$$

We may view backfitting as a method to optimize a high-dimensional (penalized) parametric problem. For example, with smoothing spline smoothers, we have seen in chapter ??? that the smoothing spline fit can be represented in terms of basis functions and we have to solve a penalized parametric problem. Backfitting can then be viewed as a coordinate-wise optimization method which optimizes one coordinate (corresponding to one predictor variable) at a time while keeping all others (corresponding to all other predictors) fixed. This coordinate-wise optimization may be slow but backfitting is a very general overall method which directly allows to use one-dimensional smoothing algorithms.

7.2.2 Additive model fitting in R

Additive models can be fitted in R with the function **gam** (**g**eneralized **a**dditive **m**odel) from the library **modreg**. The term "generalized" allows also to fit additive logistic regression, among other things.

The function **gam** uses for the smoothers $\mathcal{S}_j$ a penalized regression spline: i.e. a spline with selected knots including a penalty term (this is somewhat different than a smoothing spline). Interestingly, the function will choose the degrees of freedom, which may be different for every fitted function $\hat{g}_j(\cdot)$, via generalized cross-validation: in particular, this allows to use more degrees of freedom for "complex" functions and few degrees of freedom for functions which seem "simple".

We consider as an example the daily ozone concentration in the Los Angeles basin as a function of 9 predictor variables.

The commands and output in R look as follows.

```
library(mgcv)
ozone.dat <-
read.table("/u/sfs/Archives-Data/Hastie-Tibsh.-GAM/ozone.dat",header=T)
attach(ozone.dat)

fit <-
gam(ozone~s(vh)+s(wind)+s(humidity)+s(temp)+s(ibh)+s(dpg)+s(ibt)+s(vis)+s(doy))

summary(fit)

Family: gaussian
Link function: identity

Formula:
ozone ~ s(vh) + s(wind) + s(humidity) + s(temp) + s(ibh) + s(dpg) +
    s(ibt) + s(vis) + s(doy)

Parametric coefficients:
           Estimate  std. err.    t ratio    Pr(>|t|)
constant     11.776     0.1994      59.06    < 2.22e-16

Approximate significance of smooth terms:
                   edf       chi.sq     p-value
      s(vh)          1       8.6483     0.0032740
    s(wind)          1       4.0616     0.043874
s(humidity)          1       9.8555     0.0016933
    s(temp)      5.128       34.172     2.5311e-06
     s(ibh)      3.194       4.6316     0.22396
     s(dpg)      3.366       44.135     2.4551e-09
     s(ibt)      1.583       3.2223     0.14121
     s(vis)      2.148       16.181     0.00037471
     s(doy)       4.05       78.536     3.8614e-16

R-sq.(adj) = 0.796   Deviance explained =   81%
GCV score = 14.123   Scale est. = 13.119    n = 330
```

The column `edf` shows the estimated (via GCV) degrees of freedom: if they are equal to 1, a linear straight line is fitted. We get an indication about the relevance of a predictor either from Figure 7.2 or from the P-values in the summary output from `R`.

The fit of the model can be checked via residual analysis. The Tukey-Anscombe plot indicates heteroscedastic errors. We thus try the log-transform for the response variable and re-do the whole analysis. The results are given below.

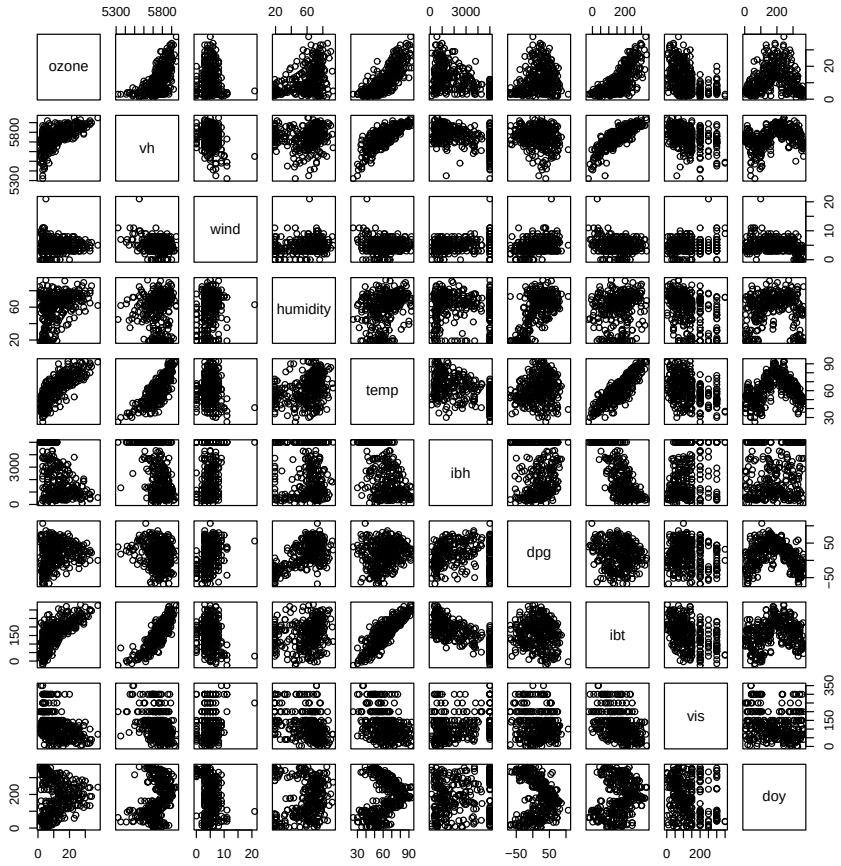


Figure 7.1: Daily ozone concentration in the Los Angeles basin as a function of 9 predictor variables.

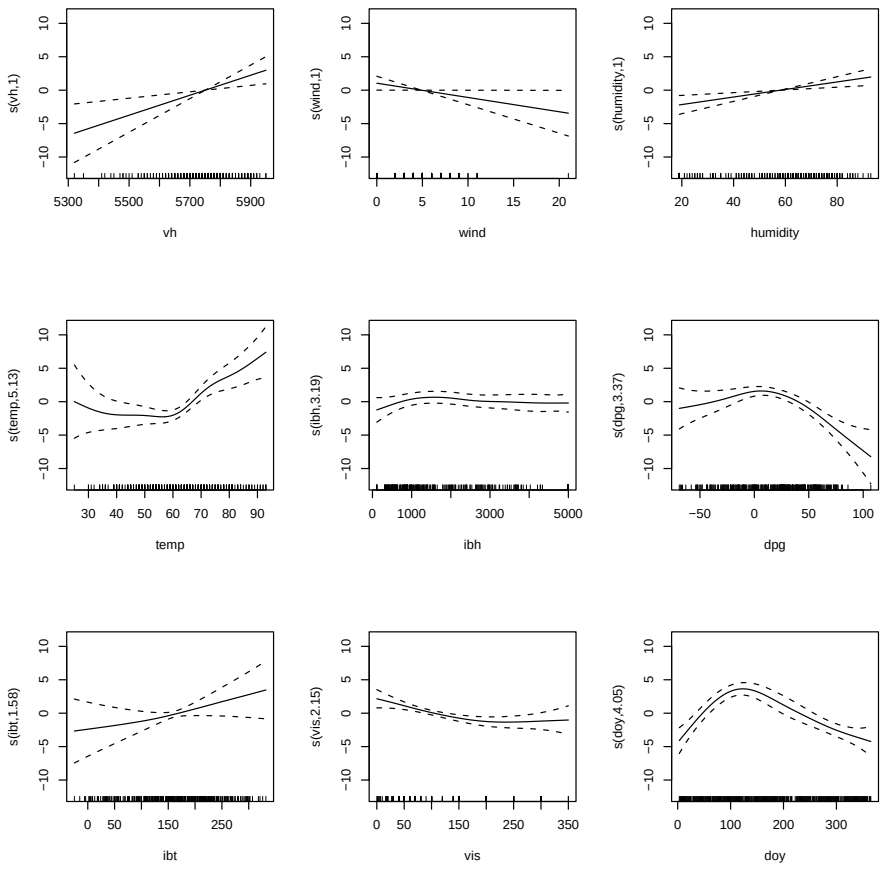


Figure 7.2: Estimated function $\hat{g}_j$ for the ozone data set.

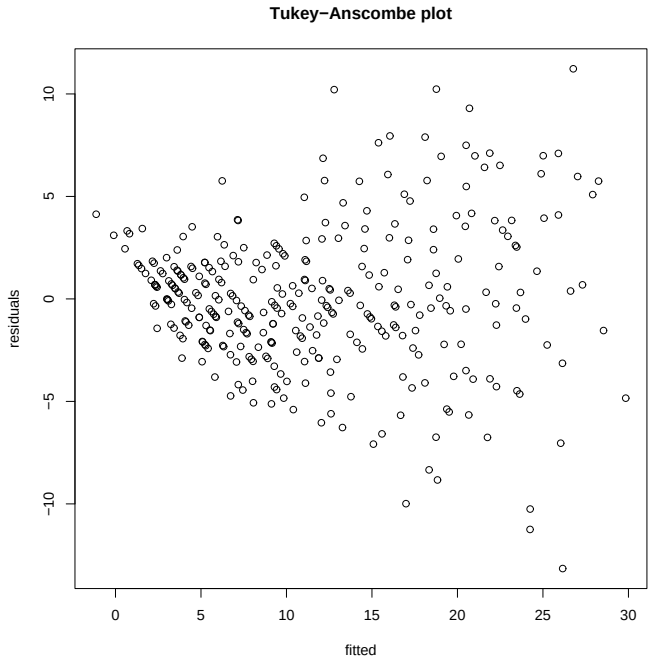


Figure 7.3: Tukey-Anscombe plot for the ozone data set.

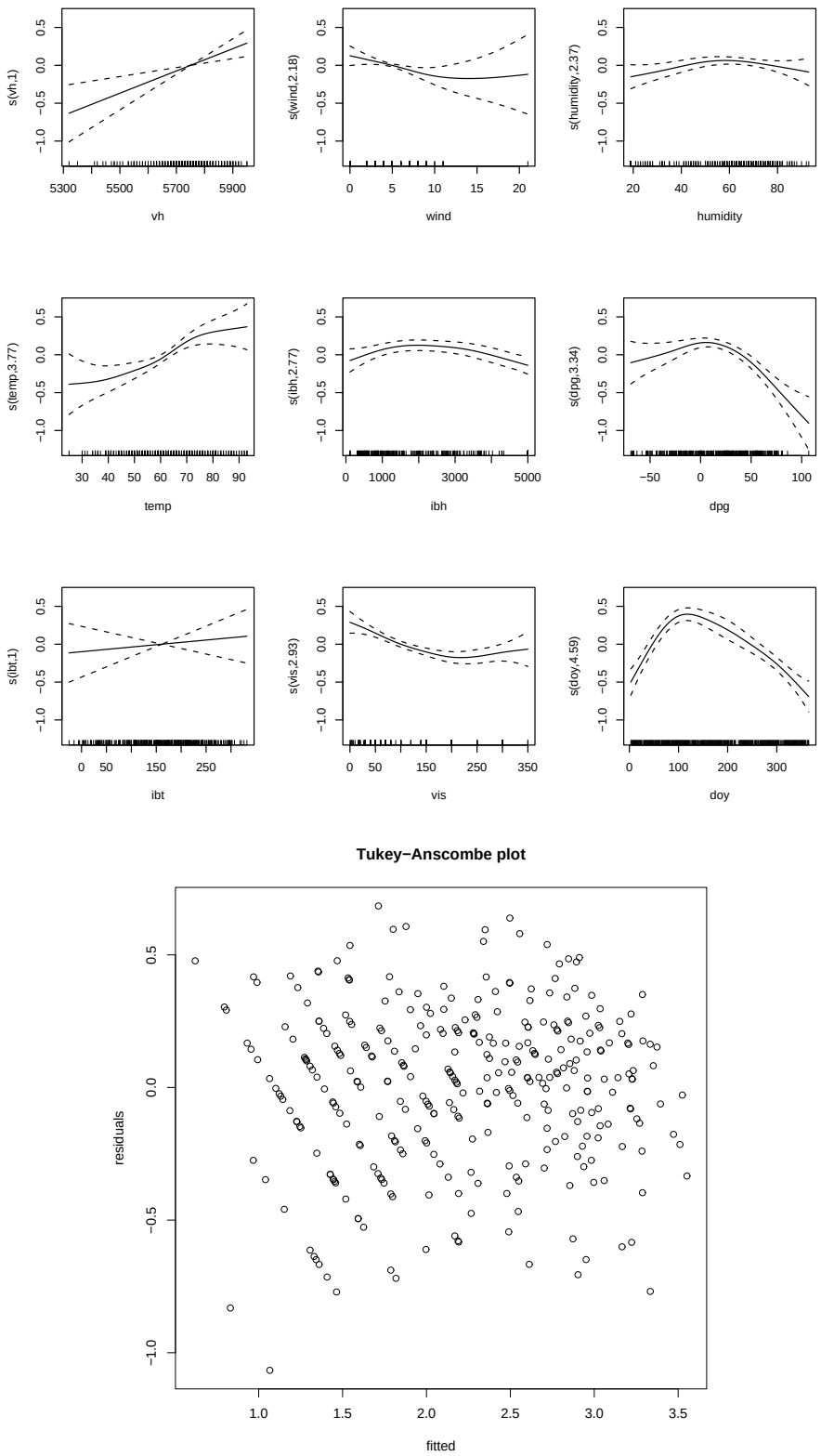


Figure 7.4: Estimated additive functions and Tukey-Anscombe plot for the ozone dataset with `log(ozone)` as the response variable.

7.3 MARS

MARS is a shortcut for **m**ultivariate **a**daptive **r**egression **s**plines. MARS is an adaptive procedure for regression and it is often useful for high-dimensional problems with many predictor variables.

MARS uses expansions in piecewise linear basis functions of the form

$$(x_j - d)_+ = \begin{cases} x - d & \text{if } x_j > d, \\ 0 & \text{otherwise.} \end{cases}$$

and $(d - x_j)_+$. The value d is a knot and $j \in \{1, \ldots, p\}$ is a component of $x \in \mathbb{R}^p$. The pair $(x_j - d)_+$, $(d - x_j)_+$ is called a *reflected pair*. The collection of reflected pairs of basis functions is then

$$\mathcal{B} = \{(x_j - d)_+, (d - x_j)_+; d \in \{x_{1j}, x_{2j}, \ldots, x_{nj}\},\ j \in \{1, 2, \ldots, p\}\}.$$

Note that $(x_j - d)_+$ is function over $\mathbb{R}^p$ but where only the jth component of $x \in \mathbb{R}^p$ is relevant.

MARS employs now a forward selection of reflected pairs of basis functions in $\mathcal{B}$ and their products. The model has the form

$$g(x) = \mu + \sum_{m=1}^{M} \beta_m h_m(x),$$

where each function $h_m(\cdot)$ is a function from the basis $\mathcal{B}$ or a product of functions from $\mathcal{B}$. The model building technique going in a forward way as follows:

1. Start with the function $h_0(x) \equiv 1$. Initialize the model set $\mathcal{M} = \{h_0(\cdot) \equiv 1\}$. Fit the function h_0 by least squares regression, yielding the estimate $\hat{\mu} = n^{-1} \sum_{i=1}^{n} Y_i$.

2. For $r = 1, 2, \ldots$ do the following:
   Search for the pair of functions $((h_{2r-1}(\cdot), h_{2r}(\cdot))$ which are of the form

   $$\begin{aligned} h_{2r-1}(\cdot) &= h_\ell(\cdot)(x_j - d)_+, \\ h_{2r}(\cdot) &= h_\ell(\cdot)(d - x_j)_+, \end{aligned} \tag{7.1}$$

   for some h_ℓ in the model set $\mathcal{M}$, and some basis functions in $\mathcal{B}$. The best pair of functions is defined to be the one which reduces residual sum of squares most.
   The model fit is then

   $$g(x) = \hat{\mu} + \sum_{m=1}^{2r} \hat{\beta}_m h_m(\cdot),$$

   where the coefficients $\hat{\beta}_m$ are estimated by least squares.
   Enlarge the model set in every iteration (with index r) by

   $$\mathcal{M} = \mathcal{M}_{old} \cup \{(x_j - d)_+, (d - x_j)_+\},$$

   with the selected reflected pairs from (7.1).

3. Iterate step 2 until a large enough number of basis functions $h_m(\cdot)$ has been fitted.

4. Do backward deletion of pairs of functions $h_{2r-1}(\cdot), h_{2r}(\cdot)$ which increase the residual sum of squares the least.

5. Stop the backward deletion by optimizing a GCV score.

For example, the trace of solutions could look as follows:

$$
\begin{aligned}
&h_0(x) = 1,\ \mathcal{M} = \{1\},\\
&h_1(x) = (x_2 - x_{27})_+,\ h_2(x) = (x_{27} - x_2)_+,\ \mathcal{M} = \{1, (x_2 - x_{27})_+, (x_{27} - x_2)_+\},\\
&h_3(x) = (x_{27} - x_2)_+ \cdot (x_1 - x_{51})_+,\ h_4(x) = (x_{27} - x_2)_+ \cdot (x_{51} - x_1)_+,\\
&\qquad \mathcal{M} = \{1, (x_2 - x_{27})_+, (x_{27} - x_2)_+, (x_1 - x_{51})_+, (x_{51} - x_1)_+\}\\
&\ldots
\end{aligned}
$$

7.3.1 Hierarchical interactions and constraints

It becomes clear from the definition of MARS, that the algorithm proceeds hierarchically in the sense that a d-order interaction can only enter the model when an interaction of degree $d-1$ involving one predictor less is already in the model. For example, an interaction of degree 4 involving x_2, x_4, x_7, x_9 enters the model because an interaction of degree 3 involving the predictors x_2, x_4, x_7 is already in the model.

Quite often it is useful to restrict interactions to degree 2 or 3. Sometimes, we may even restrict the interaction degree to 1: MARS then yields an additive model where the predictor variables and the piecewise linear spline basis functions are included in a forward-backward adaptive way.

7.3.2 MARS in R

MARS is implemented in R in the function **mars** from the library **mda**. The syntax is as follows:

```
fit <- mars(x, y, degree=2)
#x is the n*p design matrix; y is the response vector

predict(fit,x=xnew)
#xnew is a new (test) set of predictors of dimension ntest*p
```

7.4 Neural Networks

Neural networks have been very popular in the 90's in the machine learning and artificial intelligence communities. From a statistical perspective, they can be viewed as high-dimensional nonlinear regression models.

We will discuss first feedforward neural nets with one hidden layer and one output. The model is then

$$g(x) = f_0\left(\alpha + \sum_{k=1}^{q} w_k \phi(\alpha_k + \sum_{j=1}^{p} w_{jk} x_j)\right). \tag{7.2}$$

The function $f_0(\cdot)$ is often chosen as the identity. The function $\phi(\cdot)$ is usually the sigmoid function

$$\phi(x) = \frac{\exp(x)}{1 + \exp(x)}.$$

The $w_k, w_{jk}, \alpha, \alpha_k$ are unknown parameters.

The so-called input layer consists of the p predictor variables; the values $w_k\phi(\alpha_k + \sum_{j=1}^{p} w_{jk}x_j)$ $(k = 1, \ldots, q$ make the so-called hidden layer with q units. And the so-called output is for univariate regression just a single unit.

A useful variant of (7.2) is a model which includes a linear regression component:

$$g(x) = f_0\left(\alpha + \sum_{j=1}^{p} w_{j,lin}x_j + \sum_{k=1}^{q} w_k\phi(\alpha_k + \sum_{j=1}^{p} w_{jk}x_j)\right). \tag{7.3}$$

7.4.1 Fitting neural networks in R

Feedforward neural nets can be fitted in R with the function **nnet** from the library **nnet**.

In practice,it is very important to center all the predictor variables so that they are approximately on the same. This avoids, that gradient methods for optimizing the likelihood get stuck in the "flat regions" of the sigmoid functions.

```
library(nnet)

ozone.dat <-
read.table("/u/sfs/Archives-Data/Hastie-Tibsh.-GAM/ozone.dat",header=T)
attach(ozone.dat)

#center all the predictros

model <- ozone ~ I(vh - mean(vh)) + I(wind - mean(wind)) +
  I(humidity - mean(humidity)) + I(temp - mean(temp)) + I(ibh - mean(ibh)) +
  I(dpg - mean(dpg)) + I(ibt - mean(ibt)) + I(vis - mean(vis)) +
  I(doy - mean(doy))

set.seed(22)
fit <- nnet(model, data=ozone.dat, size=3, skip=T, linout=T)


summary(fit)
a 9-3-1 network with 43 weights
options were - skip-layer connections  linear output units
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1
  -0.28    0.40    0.69    0.16    0.60   -0.43    0.40    0.76   -0.69    0.47
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2
2166.55   12.02 2022.51   37.54 -247.10   -1.85  -61.52   -8.39   44.95    2.29
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3
  -0.39   -0.20    0.05    0.42    0.38    0.59   -0.05    0.69   -0.40   -0.40
   b->o   h1->o   h2->o   h3->o   i1->o   i2->o   i3->o   i4->o   i5->o   i6->o
  13.86   -0.18   -1.14   -2.83    0.00    0.12    0.08    0.25    0.00   -0.01
  i7->o   i8->o   i9->o
   0.03   -0.01   -0.01


#without linear model component: skip=F
```

```
set.seed(22)
fit1 <- nnet(model, data=ozone.dat, size=3, skip=F,linout=T)

summary(fit1)
a 9-3-1 network with 34 weights
options were - linear output units
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
  0.12 -46.65   0.75   7.80  -0.80 -28.54  12.20  -7.86   3.42 -18.87
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2
  0.93  18.30  -0.11  -3.24  -1.91  -3.40  -4.83  -1.95 -27.50 -17.47
 b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3
 -0.39  -0.20   0.05   0.42   0.38   0.59  -0.05   0.69  -0.40  -0.40
  b->o  h1->o  h2->o  h3->o
 10.06  -2.19   8.39  -3.68
```

linout=T indicates that the function is fitted on the linear regression scale; for classification, we use linout=F (default). size=3 indicates that there will be 3 hidden units: this is a tuning parameter. skip=T enforces a neural net with a linear model component. The random seed is stored because a neural net uses per default random starting values for the high-dimensional optimization.

7.5 Projection pursuit regression

Projection pursuit regression (for regression problems) bears some similarities to feed-forward neural networks. Instead of taking a linear combination of q different sigmoid function outputs (from the q hidden units in the hidden layer) in (7.2), we use the following model:

$$g_{PPR}(x) = \mu + \sum_{k=1}^{q} f_r(\sum_{j=1}^{p} \alpha_{jk} x_j),$$
$$\sum_{j=1}^{p} \alpha_{jk} = 1,\ \mathbb{E}[f_k(\sum_{j=1}^{p} \alpha_{jk} X_j)] = 0, \quad \text{for all } k.$$

The functions $f_k(\cdot) : \mathbb{R} \to \mathbb{R}$ are nonparametric and are called *ridge functions*; the linear combinations $\sum_{j=1}^{p} \alpha_{jk} x_j$ are linear projections. Note that the function $x \mapsto f_k(\sum_{j=1}^{p} \alpha_{jk} x_j)$ only varies along the direction specified by the unit vector $\alpha_k = (\alpha_{1k}, \ldots, \alpha_{pk})^T$ since $\alpha_k^T x$ is the projection of x onto α_k; hence the name "projection pursuit". The projection pursuit model typically requires much smaller q than hidden unites in a neural network, at the expense of estimating ridge functions rather than using fixed sigmoid functions in neural nets.

Estimation of projection pursuit can be done using a backfitting algorithm. Quite often, projection pursuit yields very competitive prediction performance when choosing a reasonable number of ridge functions (e.g. by optimizing a CV score).

7.5.1 Projection pursuit regression in R

The function ppr from the library mda in R can be used for fitting projection pursuit regression. The function in ppr re-scales the projection pursuit model to

$$g_{PPR}(x) = \mu + \sum_{k=1}^{q} \gamma_k f_k(\sum_{j=1}^{p} \alpha_{jk} x_j),$$
$$\sum_{j=1}^{p} \alpha_{jk} = 1, \ \mathrm{Var}(f_k(\sum_{j=1}^{p} \alpha_{jk} X_j)) = 1, \quad \text{for all } k.$$

Consider the ozone data set.

```
library(mda)

ozone.dat <-
read.table("/u/sfs/Archives-Data/Hastie-Tibsh.-GAM/ozone.dat",header=T)
attach(ozone.dat)

x <- ozone.dat[,2:10]
y <- ozone

fit <- ppr(x,y,nterms=4)   #nterms specifies the number of ridge functions

par(mfrow=c(2,2))
plot(fit)

predict(fit,xnew)   #xnew is a new set of predictor variables
```

The estimated ridge functions are shown in Figure 7.5.

7.6 Classification and Regression Trees (CART)

Quite different from the previous methods are the so-called tree models. CART is the most well known tree model or algorithm in the statistics community.

The underlying model function for CART is

$$g_{tree}(x) = \sum_{r=1}^{R} \beta_r \mathbf{1}_{[x \in \mathcal{R}_r]},$$

where $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$ is a partition of $\mathbb{R}^p$. Thus, the function $g(\cdot)$ is modelled as *piecewise constant*.

7.6.1 Tree structured estimation and tree representation

Parameter estimation $\hat{\beta}_1, \ldots, \hat{\beta}_R$ is easy when the partition $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$ would be given. We use

$$\hat{\beta}_r = \sum_{i=1}^{n} Y_i \mathbf{1}_{[x_i \in \mathcal{R}_r]} / \sum_{i=1}^{n} \mathbf{1}_{[x_i \in \mathcal{R}_r]}$$

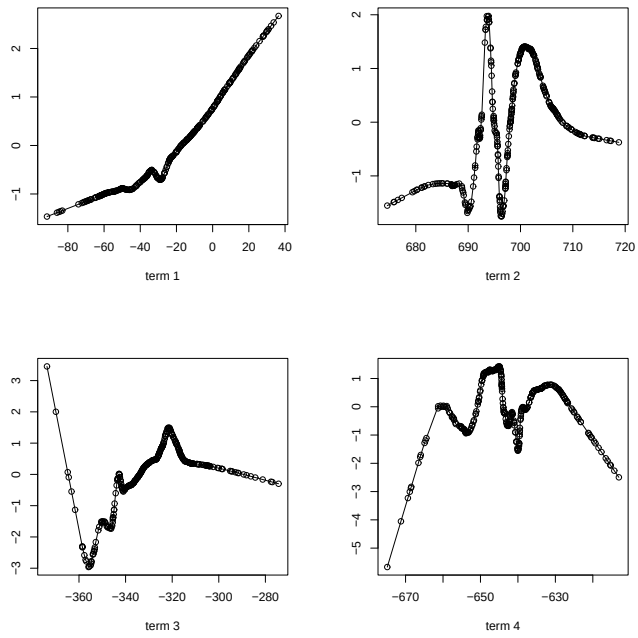


Figure 7.5: Four estimated ridge functions $f_k(\cdot)$ from a projection pursuit fit for the ozone data set.

for regression and binary classification (we do not model $g(\cdot)$ on the logistic scale). (For J class problems with $J > 2$, we an also use the tree model directly and do not need to go with a one against the rest approach, see section 6.4.2).

The tricky issue is to get a data-driven estimate for the partition $\mathcal{P}$. Some restrictions will be made to obtain a computationally feasible algorithm. This will be discussed next.

7.6.2 Tree-structured search algorithm and tree interpretation

We restrict the search for a good partition to partition cells $\mathcal{R}$ which are **axes parallel rectangles**. Moreover, we proceed in a **greedy** way since the space of partitions consisting of axes parallel rectangles is still huge.

A tree-structured greedy algorithm then proceeds as follows.

1. Start with $\mathcal{R} = 1$, $\mathcal{P} = \{\mathcal{R}\} = \mathbb{R}^p$.

2. Refine $\mathcal{R}$ into $\mathcal{R}_{left} \cup \mathcal{R}_{right}$ where:

$$\begin{aligned} \mathcal{R}_{left} &= \mathbb{R} \times \mathbb{R} \times \ldots \times (-\infty, d] \times \mathbb{R} \ldots \times \mathbb{R}, \\ \mathcal{R}_{right} &= \mathbb{R} \times \mathbb{R} \times \ldots \times (d, \infty) \times \mathbb{R} \ldots \times \mathbb{R}, \end{aligned}$$

   where one of the axes is split at the split point d, where d is from the finite set of mid-points between observed values. The search for the axes to split and the split point d are determined such that the log-likelihood is maximally reduced with the refinement (search over $j \in \{1, \ldots, p\}$ and $d \in \{\text{mid-points of observed values}\}$). Build the new partition $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2\}$ with $\mathcal{R}_1 = \mathcal{R}_{left}$, $\mathcal{R}_2 = \mathcal{R}_{right}$.

3. Refine the current partition $\mathcal{R}$ as in step 2 by refining **one** of the partition cells from the current partition $\mathcal{P}$. That is, we search for the best partition cell to refine which

includes a search as in step 2 for the best axes to split ad the best split point. Then, we up-date the partition:

$\mathcal{P} = \mathcal{P}_{old} \setminus$ partition cell selected to be refined $\cup$ {refinement cells $\mathcal{R}_{left}, \mathcal{R}_{right}$}.

4. Iterate step 3 for a large number of partition cells.

5. Backward deletion: prune the tree (see below) until a reasonable model size, typically determined via cross-validation, is achieved.

Tree representation

The search algorithm above has a useful tree representation. Figures 7.6 and 7.7 shows for part of the kyphosis dataset (2 class problem but now only with two predictor variables instead of 3) how $\mathbb{R}^2$ is recursively partitioned in a tree-structured way. As a side result, we obtain a very useful interpretation of the model in terms of a tree! Such a decision tree

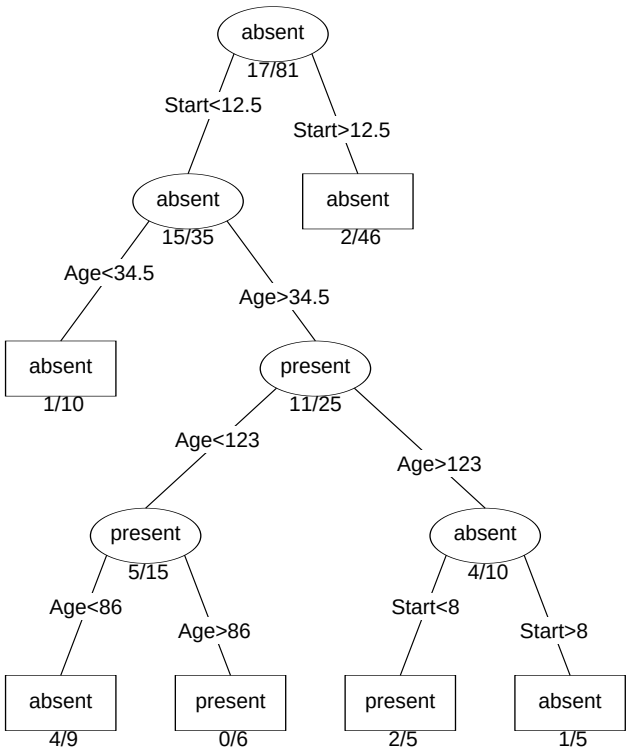


Figure 7.6: Classification tree for part of the kyphosis data set. "a" and "b" denote the frequencies of "presence" and "absence" of kyphosis in each node, and the probability estimate for kyphosis in a node is the $a/(a+b)$.

corresponds to a **recursive** partition scheme as shown in Figure 7.7.

The process of backward deletion in step 5 can now be more easily understood. Steps 1–4 result in a large tree τ_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

$$\tau_M \supset \tau_{M-1} \supset \ldots \supset \mathcal{R}_0 = \text{root tree} = \mathbb{R}^p,$$

and we can select the tree which has the best CV score.

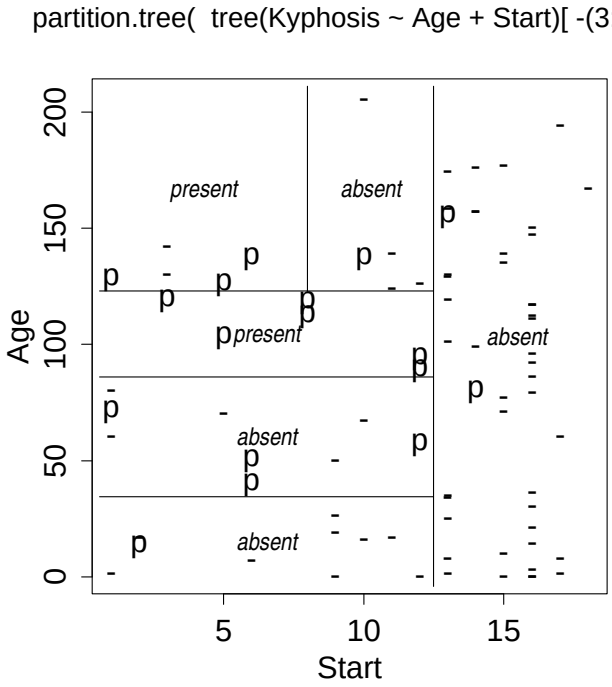


Figure 7.7: Partition of $\mathbb{R}^2$ for part of the kyphosis data set. "p" denotes presence of kyphosis, "-" denotes absence. Majority voting in each rectangle then determines the classification rule.

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. 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 isn't 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.

7.6.4 CART in R

The function `rpart` from the library `rpart` in R can be used for fitting classification or regression trees.

As an example, we run an regression tree for the ozone data set.

```
ozone.dat <-
read.table("/u/sfs/Archives-Data/Hastie-Tibsh.-GAM/ozone.dat",header=T)
attach(ozone.dat)
```

```
fit <- rpart(ozone~.,data=ozone.dat,method="anova")
             #method = ‘‘anova’’ specifies a regression tree

> fit
n= 330

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

 1) root 330 21115.4100 11.775760
   2) temp< 67.5 214  4114.3040  7.425234
     4) ibh>=3573.5 108   689.6296  5.148148 *
     5) ibh< 3573.5 106  2294.1230  9.745283
      10) dpg< -9.5 35   362.6857  6.457143 *
      11) dpg>=-9.5 71  1366.4790 11.366200
        22) ibt< 159 40   287.9000  9.050000 *
        23) ibt>=159 31   587.0968 14.354840 *
   3) temp>=67.5 116  5478.4400 19.801720
     6) ibt< 226.5 55  1276.8360 15.945450
      12) humidity< 59.5 10   167.6000 10.800000 *
      13) humidity>=59.5 45   785.6444 17.088890 *
     7) ibt>=226.5 61  2646.2620 23.278690
      14) doy>=280.5 8   398.0000 16.000000 *
      15) doy< 280.5 53  1760.4530 24.377360
        30) vis>=55 36  1149.8890 22.944440 *
        31) vis< 55 17   380.1176 27.411760 *

plot(fit)            #plotting the tree
text(fit, use.n=TRUE)
```

The fitted tree is displayed in Figure 7.8.

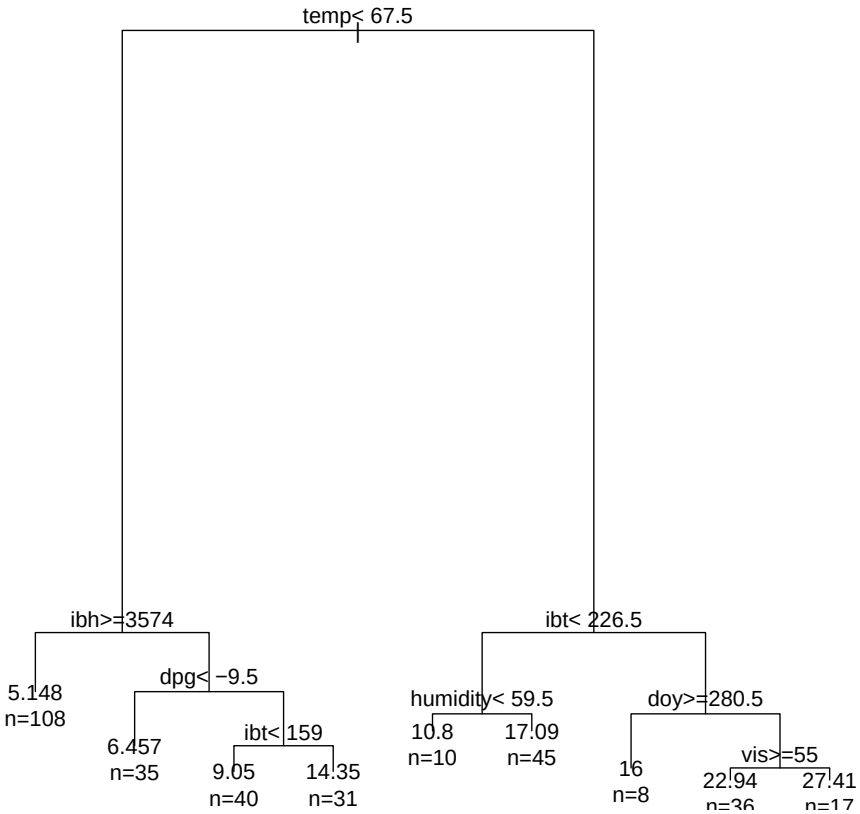


Figure 7.8: Regression tree for the ozone data set.

Chapter 8

Bagging and Boosting

8.1 Introduction

Bootstrap **agg**regat**ing** (bagging) and boosting are useful techniques to improve the predictive performance of tree models. Boosting may also be useful in connection with many other models, e.g. for additive models with high-dimensional predictors; whereas bagging is most prominent for improving tree algorithms.

8.2 Bagging

Consider a base procedure, e.g. a tree algorithm such as CART, which yields a function estimate

$$\hat{g}(\cdot) : \mathbb{R}^p \to \mathbb{R}$$

(or $\hat{g}(\cdot)$ takes values in $[0, 1]$ for classification).

8.2.1 The bagging algorithm

Bagging works as follows.

1. Generate a bootstrap sample

   $$(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$$

   and compute the bootstrapped estimator $\hat{g}^*(\cdot)$.

2. Repeat step 1 B times, yielding

   $$\hat{g}^{*1}(\cdot), \ldots, \hat{g}^{*B}(\cdot).$$

3. Aggregate the bootstrap estimates

   $$\hat{g}_{Bag}(\cdot) = B^{-1} \sum_{i=1}^{B} \hat{g}^{*i}(\cdot).$$

The bagging algorithm is nothing else than an approximation

$$\hat{g}_{Bag}(\cdot) \approx \mathbb{E}^*[\hat{g}^*(\cdot)]$$

which can be made arbitrarily good by increasing B. The novel point is that we should use now $\mathbb{E}^*[\hat{g} * (\cdot)]$ as a new estimator.

A trivial identity hints at some properties of bagging: write (the theoretical version of bagging with $B = \infty$)

$$\begin{aligned} \hat{g}_{Bag}(\cdot) &= \hat{g}(\cdot) + (\mathbb{E}^*[\hat{g}^*(\cdot)] - \hat{g}(\cdot)) \\ &= \hat{g}(\cdot) + \text{ bootstrap bias estimate.} \end{aligned}$$

Instead of subtracting the bootstrap bias estimate, we are adding it! What we can hope for is a variance reduction at the price of a higher bias. This turns out to be true if $\hat{g}(\cdot)$ is a tree-based estimator.

8.2.2 Bagging for trees

It can be shown that for tree-based estimators $\hat{g}(\cdot)$,

$$\mathrm{Var}(\hat{g}_{Bag}(x)) \overset{\text{asymp.}}{<} \mathrm{Var}(\hat{g}(x))$$

for very many x. Thus, bagging is a variance reduction technique. The reason for this is that a bagged tree turns out to be a product of probit functions $\Phi(d - \cdot)$ instead of indicator functions $\mathbf{1}_{[\cdot \le d]}$. This causes a variance reduction at the price of some bias. For example,

$$\mathrm{Var}(\mathbf{1}_{[X \le d]}) = \mathbb{P}[X \le d](1 - \mathbb{P}[X \le d]).$$

If $X \sim \mathcal{N}(0, 1)$ and $d = 0$, the latter quantity equals $1/2$. On the other hand:

$$\mathrm{Var}(\Phi(-X)) = \mathrm{Var}(U) = 1/12, \quad U \sim \text{Unif.}([0, 1])$$

which reduces the variance by the factor 3!

We should use large trees for bagging, because the variance reduction due to bagging asks for a large tree to balance the bias-variance trade-off.

8.2.3 Subagging

Subagging (**sub**sample **agg**regat**ing**) is a version of bagging: instead of drawing a bootstrap sample in step 1 of the bagging algorithm, we draw

$$(X_1^*, Y_1^*), \ldots, (X_m^*, Y_m^*) \text{ without replacement}$$

for some $m < n$. In some simple cases, it can be shown that $m = [n/2]$ is equivalent to bagging. Thus, subagging with $m = [n/2]$ can be viewed as a computationally cheaper version of bagging.

We consider a dataset about ozone concentration with $p = 8$ predictor variables (different from the previous ozone dataset). The performance of (su-)bagging for trees and MARS are shown in Figure 8.1. We see that bagging improves a regression tree substantially while it does not improve MARS at all (for this example).

The main drawback of bagging is the loss of interpretation in terms of a tree. It is by no means simple to interpret a linear combination of trees.

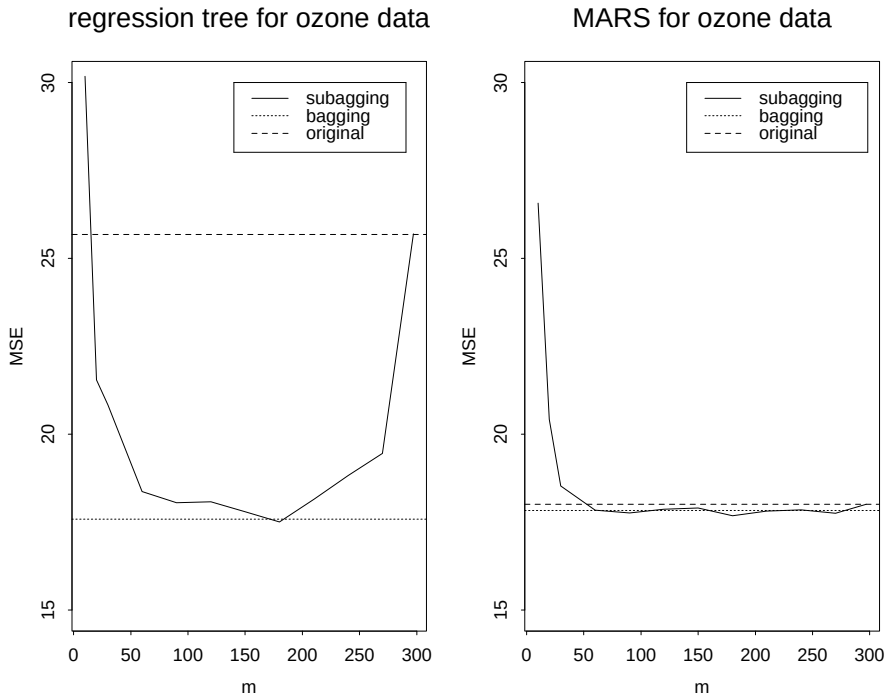


Figure 8.1: Mean squared error performance for a large regression tree and MARS and their (su-)bagged versions for an ozone data (different from the previous one).

8.3 Boosting

Boosting is a very different method to generate multiple predictions (function estimates) and combine them linearly. As with bagging, we have a base procedure yielding function estimates $\hat{g}(\cdot)$ (e.g. a tree algorithm).

8.3.1 L_2Boosting

The so-called L_2Boosting method (for regression) works as follows.

1. Fit a first function estimate from the data $\{(X_i, Y_i);\ i = 1, \ldots, n\}$ yielding a first function estimate $\hat{g}_1\cdot)$.
   Compute residuals
   $$U_i = Y_i - \hat{g}_1(X_i)\ (i = 1, \ldots, n).$$
   Denote by $\hat{f}_1(\cdot) = \nu\hat{g}_1(\cdot)\ (0 < \nu \leq 1)$.

2. For $m = 2, 3, \ldots, M$ do:
   Fit the residuals
   $$(X_i, U_i) \to \hat{g}_m(\cdot)$$
   and set
   $$\hat{f}_m(\cdot) = \hat{f}_{m-1}(\cdot) + \nu\hat{g}_m(\cdot).$$
   Compute the current residuals
   $$U_i = Y_i - \hat{f}_m(X_i)\ (i = 1, \ldots, n).$$

The shrinkage parameter ν can be chosen to be small, e.g. $\nu = 0.1$. The stopping parameter M is a tuning parameter of boosting. For ν small we typically have to choose M large.

Boosting is a bias reduction technique, in contrast to bagging. Boosting typically improves the performance of a single tree model. A reason for this is that we often cannot construct trees which are sufficiently large due to thinning out of observations in the terminal nodes. Boosting is then a device to come up with a more complex solution by taking linear combination of trees. In presence of high-dimensional predictors, boosting is also very useful as a regularization technique for additive or interaction modeling.