Nonparametric Regression\footnote{An excerpt of the lecture notes “Computational Statistics” by Peter Bühlmann and Martin Mächler}

WBL Angewandte Statistik, AS 2014

Martin Mächler

Seminar für Statistik
ETH Zürich

Sep. 2014 \hspace{1cm} (September 12, 2014)
## Contents

0 Einführung Nichtparametrische Regression für WBL ................................. 1  
  0.1 Das Lineare Modell ............................................................................. 1  
    0.1.1 Goals of a linear regression analysis ........................................... 2  
  0.2 Least Squares Method ....................................................................... 2  
    0.2.1 The normal equations ................................................................... 2  
    0.2.2 Assumptions for the Linear Model ............................................... 3  
    0.2.3 Geometrical Interpretation ............................................................. 3  
  0.3 Properties of Least Squares Estimates ............................................... 5  
    0.3.1 Moments of least squares estimates ............................................. 5  
  0.4 Rückblick Kreuzvalidierung (‘Cross-Validation’) .................................. 5  
    0.4.1 Leave-one-out cross-validation ..................................................... 5  
1 Nonparametric Density Estimation .......................................................... 7  
  1.1 Introduction ....................................................................................... 7  
  1.2 Estimation of a density ....................................................................... 7  
    1.2.1 Histogram ......................................................................................... 7  
    1.2.2 Kernel estimator .............................................................................. 8  
  1.3 The role of the bandwidth .................................................................... 9  
    1.3.1 Variable bandwidths: $k$ nearest neighbors .................................... 10  
    1.3.2 The bias-variance trade-off ............................................................ 10  
    1.3.3 Asymptotic bias and variance ......................................................... 10  
    1.3.4 Estimating the bandwidth ............................................................... 12  
  1.4 Higher dimensions .............................................................................. 13  
    1.4.1 The curse of dimensionality ........................................................... 13  
2 Nonparametric Regression ....................................................................... 15  
  2.1 Introduction ....................................................................................... 15  
  2.2 The kernel regression estimator ........................................................... 15  
    2.2.1 The role of the bandwidth ............................................................. 17  
    2.2.2 Inference for the underlying regression curve ............................... 17  
  2.3 Local polynomial nonparametric regression estimator ......................... 19  
  2.4 Smoothing splines and penalized regression ......................................... 20  
    2.4.1 Penalized sum of squares ............................................................... 20  
    2.4.2 The smoothing spline solution ....................................................... 20  
    2.4.3 Shrinking towards zero ................................................................. 21  
    2.4.4 Relation to equivalent kernels ....................................................... 21
3 Flexible regression and classification methods

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>Additive models</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Backfitting for additive regression models</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Additive model fitting in R</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>MARS</td>
<td>26</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Hierarchical interactions and constraints</td>
<td>29</td>
</tr>
<tr>
<td>3.3.2</td>
<td>MARS in R</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Neural Networks</td>
<td>29</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Fitting neural networks in R</td>
<td>30</td>
</tr>
<tr>
<td>3.5</td>
<td>Projection pursuit regression</td>
<td>31</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Projection pursuit regression in R</td>
<td>32</td>
</tr>
</tbody>
</table>
Einführung Nichtparametrische Regression für WBL

Ein kurzer Rückblick darauf, was wir für den Block “Nichtparametrische Regression” voraussetzen, inklusive der verwendeten Notation. Das Folgende sei nur Referenz für frühere WBL Blöcke und als Hilfe für eine konsistente Notation.

0.1 Das Lineare Modell

Modell Schreibweise:

\[
Y_i = \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i \quad (i = 1, \ldots, n)
\]

Wir nehmen normalerweise an, \(\varepsilon_1, \ldots, \varepsilon_n\) i.i.d. (independent, identically distributed) sind, mit \(\mathbb{E}[\varepsilon_i] = 0, \ Var(\varepsilon_i) = \sigma^2\).

Notation — deutsch und englisch:

- \(Y = \{Y_i; i = 1, \ldots, n\}\) ist der Vektor der Zielvariable / \textbf{E}: the vector of the response variables
- \(x^{(j)} = \{x_{ij}; i = 1, \ldots, n\}\) ist der Vektor der \(j\)-ten Kovariable / \textbf{E}: predictor (covariable), \((j = 1, \ldots, p)\)
- \(x_i = \{x_{ij}; j = 1, \ldots p\}\) der Predictor Vektor für die \(i\)-te Beobachtung / \textbf{E}: the vector of predictors for the \(i\)th observation \((i = 1, \ldots, n)\)
- \(\beta = \{\beta_j; j = 1, \ldots, p\}\) der Vektor der unbekannten Parameter / \textbf{E}: the vector of the unknown parameters
- \(\varepsilon = \{\varepsilon_i; i = 1, \ldots, n\}\) ist der Vektor der (zufälligen) Fehler / \textbf{E}: the vector of the random errors
- \(n\) die Stichprobengröße / \textbf{E}: the sample size,
  \(p\) die Anzahl Kovariable / \textbf{E}: the number of predictors

Modell in Vektor Schreibweise:

\[
Y_i = x_i^\top \beta + \varepsilon_i \quad (i = 1, \ldots, n)
\]
Modell in Matrix Form:

\[
\begin{pmatrix}
Y \\
n \times 1
\end{pmatrix} =
\begin{pmatrix}
X \\
n \times p
\end{pmatrix} \times
\begin{pmatrix}
\beta \\
p \times 1
\end{pmatrix} +
\begin{pmatrix}
\varepsilon \\
n \times 1
\end{pmatrix}
\]

(2)

wobei \(X\) eine \((n \times p)\)-Matrix mit Zeilen \(x_i^\top\) und Spalten \(x^{(j)}\) ist.

Die erste Kovariable (predictor variable) ist meistens konstant, d.h., \(x_{i1} \equiv 1\) für alle \(i\).

Somit haben wir einen “Intercept” (“Achsenabschnitt”, “Konstante”) im Modell:

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

Wir nehmen typischerweise an, dass der Stichprobenumfang \(n\) grösser ist als die Anzahl Kovariablen \(p\), \(n > p\), und zudem, dass die Matrix \(X\) vollen Rang \(p\) hat, d.h. dass die \(p\) Spaltenvektoren \(x^{(1)}, \ldots, x^{(p)}\) linear unabhängig sind.

0.1.1 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 0.2).

• **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 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, recall model selection.

0.2 Least Squares Method

We assume the linear model \(Y = X\beta + \varepsilon\). We are looking for a “good” estimate of \(\beta\). The least squares estimator \(\hat{\beta}\) is defined as

\[
\hat{\beta} = \arg \min_\beta \|Y - X\beta\|^2,
\]

(3)

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

0.2.1 The normal equations

The minimizer in (3) can be computed explicitly (assuming that \(X\) has rank \(p\)). Computing partial derivatives of \(\|Y - X\beta\|^2\) with respect to \(\beta\) (\(p\)-dimensional gradient vector), evaluated at \(\hat{\beta}\), and setting them to zero yields

\[
(-2) \ X^\top(Y - X\hat{\beta}) = 0 \quad ((p \times 1) - \text{null-vector}).
\]
Thus, we get the normal equations
\[ X^\top X \hat{\beta} = X^\top Y. \] (4)

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

Assuming that the matrix \( X \) has full rank \( p \), the \( p \times p \) matrix \( X^\top X \) is invertible, the least squares estimator is unique and can be represented as
\[
\hat{\beta} = (X^\top X)^{-1}X^\top Y.
\]

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 \( X^\top X \). ¹

From the residuals \( r_i = Y_i - x_i^\top \hat{\beta} \), the usual estimate for \( \sigma^2 \) is
\[
\hat{\sigma}^2 = \frac{1}{n - p} \sum_{i=1}^{n} r_i^2.
\]
Note that the \( r_i \)'s are estimates for \( \varepsilon_i \)’s; hence the estimator is plausible, up to the somewhat unusual factor \( 1/(n - p) \). It will be shown in section 0.3.1 that due to this factor, \( \mathbb{E}[\hat{\sigma}^2] = \sigma^2 \).

### 0.2.2 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 \( x_i \)'s are exact.** This means that we can observe them perfectly.

3. **The variance of the errors is constant (“homoscedasticity”).** This means: \( \text{Var}(\varepsilon_i) = \sigma^2 \) for all \( i \).

4. **The errors are uncorrelated.** This means: \( \text{Cov}(\varepsilon_i, \varepsilon_j) = 0 \) for all \( i \neq j \).

5. **The errors \( \{\varepsilon_i; \, i = 1, \ldots, n\} \) are jointly normally distributed.** This implies that also \( \{Y_i; \, i = 1, \ldots, n\} \) are jointly normally distributed.

### 0.2.3 Geometrical Interpretation

The response variable \( Y \) 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 \( Y \) with respect to the Euclidean distance. But this means geometrically that
\[
X\hat{\beta} \text{ is the orthogonal projection of } Y \text{ onto } \mathcal{X}.
\]

¹Let \( X = QR \) with orthogonal \( (n \times p) \) matrix \( Q \) and upper (Right) triangular \( (p \times p) \) \( R \). Because of \( X^\top X = R^\top Q^\top Q R = R^\top R \), computing \( \hat{\beta} \) only needs subsequent solution of two triangular systems: First solve \( R^\top c = X^\top Y \) for \( c \), and then solve \( R\hat{\beta} = c \). Further, when \( \text{Cov}(\hat{\beta}) \) requires \( (X^\top X)^{-1} \), the latter is \( R^{-1}(R^{-1})^\top \).
We denote the (vector of) fitted values by
\[ \hat{Y} = X \hat{\beta}. \]
They can be viewed as an estimate of \( X\beta \).
The (vector of) residuals is defined by
\[ \mathbf{r} = Y - \hat{Y}. \]
Geometrically, it is evident that the residuals are orthogonal to \( \mathcal{X} \), because \( \hat{Y} \) is the orthogonal projection of \( Y \) onto \( \mathcal{X} \). This means that
\[ \mathbf{r}^\top \mathbf{x}^{(j)} = 0 \text{ for all } j = 1, \ldots, p, \]
where \( \mathbf{x}^{(j)} \) is the \( j \)th column of \( X \).
We can formally see why the map
\[ Y \mapsto \hat{Y} \]
is an orthogonal projection. Since \( \hat{Y} = X \hat{\beta} = X(X^\top X)^{-1}X^\top Y \), the map can be represented by the matrix
\[ P = X(X^\top X)^{-1}X^\top. \]
(5)
It is evident that \( P \) is symmetric (\( P^\top = P \)) and \( P \) is idempotent (\( P^2 = P \)). Furthermore
\[ \sum_i P_{ii} = \text{tr}(P) = \text{tr}(X(X^\top X)^{-1}X^\top) = \text{tr}((X^\top X)^{-1}X^\top X) = \text{tr}(I_{p \times p}) = p. \]
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 \( \mathcal{X}^\perp \).
0.3 Properties of Least Squares Estimates

0.3.1 Moments of least squares estimates

We assume here the usual linear model
\[ Y = X\beta + \varepsilon, \quad E[\varepsilon] = 0, \quad \text{Cov}(\varepsilon) = E[\varepsilon\varepsilon^T] = \sigma^2 I_{n\times n}. \] (6)

This means that assumption 1.-4. from section 0.2.2 are satisfied.

It can then be shown that:

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

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

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

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

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

\[ \text{Var}(r_i) = \sigma^2 (1 - P_{ii}). \]

From this, we obtain

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

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

0.4 Rückblick Kreuzvalidierung (‘Cross-Validation’)

Wir setzen hier voraus, dass Sie noch wissen, was Kreuzvalidierung ( / E: “Cross validation”, auch “CV”) bedeutet, da diese für die nichtparametrische Regression ein entscheidend wichtiges Werkzeug ist.

Das Ziel der Kreuzvalidierung ist es, ein “realistisches” Gütemass für die Vorhersage-Genauigkeit (“prediction accuracy”) eines statistischen Modells (z.B. Parametervektor \( \theta \)) zu erhalten.

Hier eine kondensierte “Definition” der einfachsten Art von Kreuzvalidierung; allgemein formuliert, aber mit Fokus auf den Fall der nichtparametrischen Regression, wo das geschätzte Modell mit \( \hat{m}(x) \) bezeichnet wird.

0.4.1 Leave-one-out cross-validation

For leave-one-out cross-validation (l.o.o. CV), we use the \( i \)th 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 \( i \)th sample, the estimator is based on the sample without the \( i \)th observation, and we denote this by

\[ \hat{\theta}_n^{(-i)}, \quad i = 1, \ldots, n. \]

We can then evaluate this estimate on the \( i \)th 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 \( \rho \), e.g. \( \rho(u) = u^2 \) as in Chapter 2. The cross-validated performance is then

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

which is an estimate of the test set error, or generalization error, see the topic “Cross Validation”.

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

CV requires that the estimator, e.g. here \( \hat{m}(.) \), is fitted \( n \) times, namely for all training sets where the \( i \)th observation has been deleted \( (i = 1, \ldots, n) \).
Chapter 1

Nonparametric Density Estimation

1.1 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 2) and flexible regression and classification methods with many predictor variables (Chapter 3).

1.2 Estimation of a density

We consider the data which records the duration of eruptions of “Old Faithful”, a famous geyser in Yellowstone National Park (Wyoming, USA). You can see pictures or watch it via webcam on \url{http://www.nps.gov/yell/}.

1.2.1 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

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

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 1.1: Histograms (different class widths) for durations of eruptions of “Old Faithful” geyser in Yellowstone Park \((n = 272, \text{data(faithful)})\).

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.

1.2.2 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

\[
f(x) = \lim_{h \to 0} \frac{1}{2h} \mathbb{P}[x - h < X \leq x + h]. \tag{1.1}
\]

The naive estimator is then constructed without taking the limit in (1.1) and by replacing probabilities with relative frequencies:

\[
\hat{f}(x) = \frac{1}{2hn} \#\{i; X_i \in (x - h, x + h]\}. \tag{1.2}
\]

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 (1.2) is as follows. Define the weight function

\[
w(x) = \begin{cases} 
1/2 & \text{if } |x| \leq 1, \\
0 & \text{otherwise}.
\end{cases}
\]
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
\[ \hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{x - X_i}{h} \right), \]
\[ K(x) \geq 0, \quad \int_{-\infty}^{\infty} K(x)dx = 1, \quad K(x) = K(-x). \tag{1.3} \]

The estimator depends on the bandwidth \( h > 0 \) which acts as a tuning parameter. For large bandwidth \( h \), the estimate \( \hat{f}(\cdot) \) 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.

The smoothness of \( \hat{f}(\cdot) \) is inherited from the smoothness of the kernel: if the \( r \)th 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
\[ K(x) = \varphi(x) = (2\pi)^{-\frac{1}{2}} e^{-x^2/2} \] (the density of the \( N(0,1) \) distribution)
or a kernel with finite support such as \( K(x) = \frac{3}{4} \cos(\pi x)1(|x| \leq 1) \). The Epanechnikov kernel, which is optimal with respect to mean squared error, is
\[ K(x) = \frac{3}{4} \left( 1 - |x|^2 \right) 1(|x| \leq 1). \]

But far more important than the kernel is the bandwidth \( h \), see figure 1.2: its role and how to choose it are discussed below.

\[ \begin{align*}
0.5 & \quad 1.5 & \quad 2.5 & \quad 3.5 & \quad 4.5 & \quad 5.5 \\
0.0 & \quad 0.1 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.5 & \quad 0.6 & \quad 0.7 \\
0.0 & \quad 0.1 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.5 & \quad 0.6 & \quad 0.7 \\
0.0 & \quad 0.1 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.5 & \quad 0.6 & \quad 0.7 \\
0.0 & \quad 0.1 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.5 & \quad 0.6 & \quad 0.7 \\
0.0 & \quad 0.1 & \quad 0.2 & \quad 0.3 & \quad 0.4 & \quad 0.5 & \quad 0.6 & \quad 0.7 \\
\end{align*} \]

Figure 1.2: kernel density estimates of the “Old Faithful” eruption lengths; Gaussian kernel and bandwidths \( h = 0.064, 0.102, 0.164, 0.262, 0.419 \).

1.3 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 “\( \delta \)-spikes” at every observation \( X_i \), whereas \( \hat{f}(\cdot) = \hat{f}_h(\cdot) \) becomes smoother as \( h \) is increasing.
1.3.1 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, \( \hat{f}_{h(x)}(\cdot) \) will not be a density anymore since the integral is not necessarily equal to one.

1.3.2 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 \( \text{MSE}(x) \) at a point \( x \),

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

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)

\[
\text{IMSE} = \int \text{MSE}(x) \, dx
\]

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, denoted as \( \text{MISE} \) (mean integrated squared error) and written as

\[
\text{MISE} = \mathbb{E} \left[ \int \left( \hat{f}(x) - f(x) \right)^2 \, dx \right] = \mathbb{E} \left[ \text{ISE} \right], \quad (1.4)
\]

where \( \text{ISE} = \int \left( \hat{f}(x) - f(x) \right)^2 \, dx \).

1.3.3 Asymptotic bias and variance

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

\[
\mathbb{E}[\hat{f}(x)] = \int \frac{1}{h} K \left( \frac{x - y}{h} \right) f(y) \, dy
\]

\[
\text{Var}(\hat{f}(x)) = \frac{1}{nh^2} \text{Var} \left( K \left( \frac{x - X_i}{h} \right) \right) = \frac{1}{nh^2} \mathbb{E} \left[ K \left( \frac{x - X_i}{h} \right) \right]^2 - \frac{1}{n h^2} \mathbb{E} \left[ K \left( \frac{x - X_i}{h} \right) \right]^2
\]

\[
= n^{-1} \int \frac{1}{h^2} K \left( \frac{x - y}{h} \right)^2 f(y) \, dy - n^{-1} \left( \int \frac{1}{h} K \left( \frac{x - y}{h} \right) f(y) \, dy \right)^2. \quad (1.5)
\]
For the bias we therefore get (by a change of variable and $K(-z) = K(z)$)

$$\text{Bias}(x) = \int \frac{1}{h} K \left( \frac{x - y}{h} \right) f(y)dy - f(x)$$

$$= \int_{z=(y-x)/h} K(z)f(x + hz)dz - f(x) = \int K(z) (f(x + hz) - f(x)) dz, \quad (1.6)$$

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:

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

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:

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

Plugging this into (1.6) yields

$$\text{Bias}(x) = hf'(x) \int zK(z)dz + \frac{1}{2} h^2 f''(x) \int z^2 K(z)dz + \ldots$$

$$= \frac{1}{2} h^2 f''(x) \int z^2 K(z)dz + \text{higher order terms in } h.$$  

For the variance, we get from (1.5)

$$\text{Var}(\hat{f}(x)) = n^{-1} \int \frac{1}{h^2} K \left( \frac{x - y}{h} \right)^2 f(y)dy - n^{-1} (f(x) + \text{Bias}(x))^2$$

$$= n^{-1} h^{-1} \int f(x - hz)K(z)^2 dz - n^{-1} (f(x) + \text{Bias}(x))^2$$

$$= n^{-1} h^{-1} \int f(x - hz)K(z)^2 dz + O(n^{-1}) = n^{-1} h^{-1} f(x) \int K(z)^2 dz + o(n^{-1} h^{-1}),$$

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 $n \to \infty$

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

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

$$h_{opt}(x) = n^{-1/5} \left( \frac{f(x) \int K^2(z) dz}{(f''(x))^2 (\int z^2 K(z) dz)^2} \right)^{1/5}. \quad (1.7)$$

Since it’s not straightforward to estimate and use a local bandwidth $h(x)$, one rather considers minimizing the MISE, i.e., $\int MSE(x) dx$ which is asymptotically

$$\text{asympt. MISE} = \int \text{Bias}(x)^2 + \text{Var}(\hat{f}(x)) dx = \frac{1}{4} h^4 R(f'') \sigma_k^2 + R(K)/(nh), \quad (1.8)$$
where \( R(g) = \int g^2(x) \, dx \), \( \sigma_K^2 = \int x^2 K(x) \, dx \), and the “global” asymptotically optimal bandwidth becomes
\[
h_{opt} = n^{-1/5} \left( R(K)/\sigma_K^4 \right)^{1/5}. \tag{1.9}
\]
By replacing \( h \) with \( h_{opt} \), e.g., in (1.8), we see that both variance and bias terms are of order \( O(n^{-1/5}) \), the optimal rate for the MISE and MSE\( (x) \). From section 1.4.1, this rate is also optimal for a much larger class of density estimators.

### 1.3.4 Estimating the bandwidth

As seen from (1.9), the asymptotically best bandwidth depends on \( R(f'') = \int f''^2(x) \, dx \) which is unknown (whereas as \( R(K) \) and \( \sigma_K^2 \) are known). It is possible to estimate the \( f'' \) again by a kernel estimator with an “initial” bandwidth \( h_{init} \) (sometimes called a pilot bandwidth) yielding \( \hat{f}''_{init} \). Plugging this estimate into (1.9) yields an estimated bandwidth \( \hat{h} \) for the density estimator \( \hat{f}(\cdot) \) (the original problem): of course, \( \hat{h} \) depends on the initial bandwidth \( h_{init} \), but choosing \( h_{init} \) in an ad-hoc way is less critical for the density estimator than choosing the bandwidth \( h \) itself. Furthermore, methods have been devised to determine \( h_{init} \) and \( h \) simultaneously (e.g., “Sheather-Jones”, in \( R \) using \texttt{density(\*, bw="SJ")}.

#### Estimating local bandwidths

Note that the \( h_{opt}(x) \) bandwidth selection in (1.7) is more problematical mainly because \( \hat{f}_{h_{opt}(x)}(x) \) will not integrate to one without further normalization. On the other hand, it can be important to use \textit{locally varying} bandwidths instead of a single global one in a kernel estimator at the expense of being more difficult. The plug-in procedure outlined above can be applied \textit{locally}, i.e., conceptually for each \( x \) and hence describes how to estimate local bandwidths from data and how to implement a kernel estimator with locally varying bandwidths. In the related area of nonparametric regression, in section 2.2.1, we will show an example about locally changing bandwidths which are estimated based on an iterative version of the (local) plug-in idea above.

#### Other density estimators

There are quite a few other approaches to density estimation than the kernel estimators above (whereas in practice, the \textit{fixed} bandwidth kernel estimators are used predominantly because of their simplicity). An important approach in particular aims to estimate the \textit{log density} \( \log f(x) \) (setting \( \hat{f} = \exp(\hat{\log f}) \)) which has no positivity constraints and whose “normal limit” is a simple quadratic. One good implementation is in Kooperberg’s \( R \) package \texttt{logspline}, where spline knots are placed in a stepwise algorithm minimizing approximate BIC (or AIC). This is can be seen as another version of locally varying bandwidths.
1.4 Higher dimensions

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

$$X_1, \ldots, X_n \text{ i.i.d. } \sim f(x_1, \ldots, x_d) \, dx_1 \cdots dx_d$$

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

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

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

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

$$K(u) \geq 0, \quad \int_{\mathbb{R}^d} K(u) \, du = 1, \quad \int_{\mathbb{R}^d} uK(u) \, du = 0, \quad \int_{\mathbb{R}^d} uu^\top K(u) \, du = I_d.$$

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

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

If one additionally desires the multivariate kernel $K(u)$ to be radially symmetric, it can be shown that $K$ must be the multivariate normal (Gaussian) density, $K(u) = c_d \exp\left(-\frac{1}{2} u^\top u\right)$.

1.4.1 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 $x$ 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 $d$:

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

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

Chapter 2

Nonparametric Regression

2.1 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 1.4.1 and often require different approaches, as will be discussed later in Chapter 3.

Figure 2.1 shows (several identical) scatter plots of \((x_i, Y_i)\) \((i = 1, \ldots, n)\). We can model such data as

\[
Y_i = m(x_i) + \epsilon_i,
\]

(2.1)

where \(\epsilon_1, \ldots, \epsilon_n\) i.i.d. with \(\mathbb{E}[\epsilon_i] = 0\) and \(m : \mathbb{R} \rightarrow \mathbb{R}\) 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 2.1 does not 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.

2.2 The kernel regression estimator

We can view the regression function in (2.1) 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 2.1: 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

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

(2.2)
i.e., a weighted mean of the \( Y_i \) where \( \omega_i = \omega_i(x) \) is a kernel centered at \( x_i \). An interesting interpretation of the kernel regression estimator in (2.2) is

\[
\hat{m}(x) = \arg \min_{m \in \mathbb{R}} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right) (Y_i - m)^2.
\] (2.3)

This can be easily verified by solving \( \frac{d}{d m} \sum_{i=1}^{n} K((x - x_i)/h)(Y_i - m)^2 = 0 \). 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 those observations \((x_i, Y_i)\) where \( x_i \) is close to the point \( x \) of interest.

### 2.2.1 The role of the bandwidth

Analogously as in section 1.3, 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. We are not showing the computations for \( \text{MSE}(x) \), just note that they not only depend on (derivatives of) \( m(x) \), but also on \( f_X(x) \).

#### Local bandwidth selection

Similarly as in (1.7), also using \( \int uK(u) du = 0 \), (assuming equidistant \( x_i \), i.e., constant \( x_{i+1} - x_i \)) there is a formula of the asymptotically best local bandwidth \( h_{opt}(x) \) which depends on \( m''(\cdot) \) and the error variance \( \sigma^2_x \):

\[
h_{opt}(x) = n^{-1/5} \left( \frac{\sigma^2_x \int K^2(z) dz}{\left\{ m''(x) \int (z^2 K(z) dz)^2 \right\}^{1/2}} \right)^{1/5}.
\] (2.4)

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 \( \sigma^2_x \); 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 \( \sigma^2_x \), 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 \texttt{lokerns} in the package \texttt{lokern}. The dataset \texttt{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:

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

### 2.2.2 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 \( x_i \).

---

1 The equidistant case is a simple special case (with \( f(x) \equiv 1 \)) of the general “random design case”, \( X_i \sim f(x) \, dx \), where the design density \( f() \) plays an important role: The bias, \( \mathbb{E} [\hat{m}(x)|X_1, \ldots, X_n] - m(x) \), is asymptotically, \( B(x, h) = h^2 \left( m''(x) + 2 m'(x) f'(x)/f(x) \right) \cdot \sigma^2_k / 2 + o(h^2) \), and the (conditional) variance \( V(x, h) = \frac{1}{m_n} \mathbb{R}(K)/f(x) \cdot \sigma^2_x + o \left( \frac{1}{mn} \right) \), with the same notation as in (1.8) and (1.9) for kernel density estimation, also for the heteroscedastic case where \( \sigma^2_x = \sigma^2_x(x) \).
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 (on $\mathbb{R}^n$, i.e., a matrix):

$$S : \mathbb{R}^n \to \mathbb{R}^n,$$

$$(Y_1, \ldots, Y_n)^\top \mapsto (\hat{m}(x_1), \ldots, \hat{m}(x_n))^\top =: \hat{Y},$$

i.e., $\hat{Y} = SY$ where $S$ is the matrix representing the linear operator above. The kernel estimator in (2.2) is of the form

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

Therefore, the matrix $S$ which represents the operator above is

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

since $S(Y_1, \ldots, Y_n)^\top = (\hat{m}(x_1), \ldots, \hat{m}(x_n))^\top$. The “smoother” matrix $S$ is also called the “hat matrix”, since it yields the vector of fitted values (at the observed design points $x_i$). Note that many other nonparametric regression methods (including those in the next two sections) can be seen to be linear in $Y$ and hence be written as $\hat{Y} = SY$ where $S$ depends on the $x$-design $(x_1, x_2, \ldots, x_n)$ and typically a smoothing parameter, say $h$.

Algorithmically, for $\hat{Y} = s(x, Y, h)$, the hat matrix can easily be computed columnwise as $S_{.,j} = s(x, e_j, h)$ where $e_j$ is the unit vector with $(e_j)_i = \delta_{i,j} := 1(i = j)$.

Because of the elementary formula $\text{Cov}(AX) = A \text{Cov}(X) A^\top$ (for a non-random matrix $A$ and random vector $X$), we get the covariance matrix

$$\text{Cov}(\hat{m}(x)) = \sigma_s^2 SS^\top,$$

i.e., $\text{Cov}(\hat{m}(x_i), \hat{m}(x_j)) = \sigma_s^2 (SS^\top)_{ij}$, and $\text{Var}(\hat{m}(x_i)) = \sigma_s^2 (SS^\top)_{ii}$. 

Figure 2.2: Nonparametric function estimate and locally varying bandwidths for distance of stopping as a function of speed of cars.
### Degrees of freedom

One way to assign degrees of freedom for regression estimators with a linear hat-operator $S$ is given by the trace,

$$df = \text{tr}(S).$$

(2.6)

This definition coincides with the notion we have seen in the linear model: there, (5), 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 $\text{tr}(P) = \text{tr}((X^TX)^{-1}X^T) = \text{tr}(I_p) = p$ 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 (2.5) requires knowledge of $\sigma^2_\varepsilon$. A plausible estimate is via the residual sum of squares,

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

We then get an estimate for the standard error of the kernel regression estimator at the design points via (2.5):

$$\hat{s.e}(\hat{m}(x_i)) = \sqrt{\hat{\sigma}^2_\varepsilon \left(\frac{\text{SS}}{\text{df}}\right)}.$$

(2.7)

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

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

so that

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

yields approximate pointwise confidence intervals for $\mathbb{E}[\hat{m}(x_i)]$. Some functions in R (e.g. the function `gam` from package `mgcv`, see Chapter 3) 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 $I - \text{bias}$, where bias is an estimate of the bias (which is not so easy to construct; see also section 1.3).

### 2.3 Local polynomial nonparametric regression estimator

As a starting point, consider the kernel estimator which can be represented as a locally constant function as in (2.3). This can now be extended to functions which are locally polynomial. We aim to find local regression parameters $\beta(x)$, defined as

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

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 (2.3).

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

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

2.4 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.

2.4.1 Penalized sum of squares

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

$$\sum_{i=1}^{n} (Y_i - m(x_i))^2 + \lambda \int m''(z)^2 dz, \quad (2.8)$$

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 (but for $\lambda \to 0$, in the limit, $m_\lambda \to$ the well defined interpolating natural cubic spline).
- $\lambda = \infty$: the least squares fit for linear regression which fulfills $m''(x) \equiv 0$.

Thus, a large $\lambda$ corresponds to a smooth function.

2.4.2 The smoothing spline solution

Remarkably, the minimizer of (2.8) 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. Let us assume for now that the data has $x$ values sorted and unique, $x_1 < x_2 < \ldots < x_n$.

The solution $\hat{m}_\lambda(\cdot)$ (i.e., the unique minimizer of (2.8)) 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 $\hat{m}_\lambda^{(k)}$ ($k = 0, 1, 2$) is continuous everywhere and (“natural”) $\hat{m}_\lambda''(x_1) = \hat{m}_\lambda''(x_n) = 0$. For the $n - 1$ cubic polynomials, we'd need $(n - 1) \cdot 4$ coefficients. Since there are $(n - 2) \cdot 3$ continuity conditions (at every “inner knot”, $i = 2, \ldots, n - 1$) plus the
2 “natural” conditions, this leaves \(4(n-1) - [3(n-2) + 2] = n\) free parameters (the \(\beta_j\)'s below). Knowing that the solution is a cubic spline, it can be obtained by linear algebra. The trick is to represent

\[
m(\lambda)(x) = \sum_{j=1}^{n} \beta_j B_j(x),
\]

(2.9)

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 (2.8) for \(\hat{m}(\lambda)\) as in (2.9) then becomes

\[
\|Y - B\beta\|^2 + \lambda \beta^\top \Omega \beta,
\]

where the design matrix \(B\) has \(j\)th column \((B_j(x_1), \ldots, B_j(x_n))^\top\) and \(\Omega_{jk} = \int B_j''(z) B_k''(z) dz\). The solution can then be derived in a straightforward way,

\[
\hat{\beta}_{n \times 1} = (B^\top B + \lambda \Omega)^{-1} B^\top Y.
\]

(2.10)

This can be computed efficiently using fast linear algebra, particularly when \(B\) is a banded matrix.

The fitted values are then \(\hat{Y}_i = \hat{m}(\lambda)(x_i)\) \((i = 1, \ldots, n)\), \(\hat{m}(\lambda)(x) = \sum_{j=1}^{n} \hat{\beta}_j B_j(x)\), and

\[
\hat{Y} = (\hat{Y}_1, \ldots, \hat{Y}_n)^\top = S_{\lambda} Y, \quad S_{\lambda} = B(B^\top B + \lambda \Omega)^{-1} B^\top.
\]

(2.11)

The hat matrix \(S_{\lambda}\) is here symmetric which implies elegant mathematical properties (real-valued eigen-decomposition).

2.4.3 Shrinking towards zero

At first sight, the smoothing spline solution in (2.9) looks heavily over-parameterized since we have to fit \(n\) unknown coefficients \(\beta_1, \ldots, \beta_n\). However, the solution in (2.10) 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 shrunk towards zero: i.e., for large \(\lambda\), the expression \((B^\top B + \lambda \Omega)^{-1}\) becomes small. Thus, since all the coefficients are shrunk 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 \(\lambda\) is chosen suitably.

Note that \(\lambda\) can be chosen on the scale of equivalent degrees of freedom (df): \(df = \text{tr}(S_{\lambda})\). 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 2.1.

2.4.4 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

\[
\hat{m}(\lambda)(x) \approx \sum_{i=1}^{n} w_i(x) Y_i,
\]

\[
w_i(x) = \frac{1}{nh(x) f_X(x)} K \left( \frac{x - x_i}{h(x)} \right),
\]

\[
h(x) = \lambda^{1/4} n^{-1/4} f_X(x)^{-1/4},
\]

\[
K(u) = \frac{1}{2} \exp \left( - \frac{|u|}{\sqrt{2}} \right) \sin \left( \frac{|u|}{\sqrt{2}} + \frac{\pi}{4} \right).
\]
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 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 2.1. Finally, we illustrate on an artificial dataset the advantage of smoothing splines to adapt to the density of the predictor variables. Figure 2.3 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 topic “Cross validation” 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.

![Figure 2.3](image_url)

Figure 2.3: Curve estimation for synthetic dataset ($n = 417$). Left panels: scatterplot, overlaid with curve estimates (red) and true curve (gray); Right panels: curve estimates (red) and true curve (gray).

Top: Smoothing spline with GCV-selected df ($= 9.23$); Bottom: Nadaraya-Watson kernel estimator with bandwidth chosen for good estimation of the strong oscillations of the true function (giving $df = 25.6$).
Chapter 3

Flexible regression and classification methods

3.1 Introduction

The curse of dimensionality makes it virtually impossible to estimate a regression function \( m(x) = \mathbb{E}[Y \mid X = x] \) or the probability functions \( \pi_j(x) = \mathbb{P}[Y = j \mid X = x] \) (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 \( \mathbb{E}[Y \mid X = x] \) or the logit transform in a binary classification problem \( \log(\pi(x)/(1 - \pi(x))) \) by

\[
g(\cdot) : \mathbb{R}^p \rightarrow \mathbb{R}.
\]

3.2 Additive models

Additive models require additivity of the function: the model is

\[
g_{\text{add}}(x) = \mu + \sum_{j=1}^{p} g_j(x_j), \quad \mu \in \mathbb{R}
\]

\[
g_j(\cdot) : \mathbb{R} \rightarrow \mathbb{R}, \quad \mathbb{E}[g_j(X_j)] = 0 \quad (j = 1, \ldots, p).
\]

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 \( \hat{g}_{\text{add}}(\cdot) \) have mean square error rate \( O(n^{-4/5}) \) as for estimating a single one-dimensional (\( \mathbb{R} \rightarrow \mathbb{R} \)) function.

---

1as a matter of fact, both the original Hastie-Tibshirani and the new algorithm in mgcv allow for bivariate interactions, optionally
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 $X_1, \ldots, X_n$ is denoted by the hat operator $S_j : (U_1, \ldots, U_n) \mapsto (\hat{U}_1, \ldots, \hat{U}_n)$ ($j = 1, \ldots, p$) for any response vector $(U_1, \ldots, U_n)^T$. The subscript $j$ indicates that smoothing is done against the $j$th predictor variable. For example, $S_j$ 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
   \[ \hat{g}_j = S_j(Y - \hat{\mu}1 - \sum_{k \neq j} \hat{g}_k), \]
   where $Y = (Y_1, \ldots, Y_n)^T$ and $\hat{g}_j = (\hat{g}_j(X_{1j}), \ldots, \hat{g}_j(X_{nj}))^T$. Stop the iterations if the individual functions $\hat{g}_j(\cdot)$ do not change much anymore, e.g.,
   \[ \frac{\|\hat{g}_{j,new} - \hat{g}_{j,old}\|_2}{\|\hat{g}_{j,old}\|_2} \leq \text{tol} \]
   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 2 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.

Additive models can be fitted in R with the function `gam` (generalized additive model) from package `mgcv`. The term “generalized” allows also to fit additive logistic regression, among other things.

The function `gam` uses for the smoothers $S_j$ a penalized regression spline: i.e., a spline with selected knots including a penalty term (this is somewhat different than a smoothing

---

Backfitting was the algorithm for fitting additive models when they were proposed (Hastie and Tibshirani, 1990, “Generalized Additive Models”), available in R package `gam`. The more sophisticated `gam()` function from package `mgcv` uses a different algorithm for fitting.
3.2 Additive models

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.

```r
library(mgcv)
data(ozone, package = "gss")
d.ozone ← ozone; colnames(d.ozone)[c(1,2,3,4)] ← ....... # 'better' names
pairs(d.ozone, pch = ".", gap = 0.1) # --> Scatterplot Matrix

Figure 3.1: Daily ozone concentration in the Los Angeles basin as a function of 9 predictor
variables.

fit ←
gam(O3 ~ s(vdht)+s(wind)+s(humidity)+s(temp)+s(ibht)+s(dgpg)+s(ibtp)+s(vsty)+s(day),
data = d.ozone)
summary(fit)

. . . . . . . .

Parametric coefficients:  
   Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.7758  0.1988 59.22 <2e-16 ***
```

Figure 3.1: Daily ozone concentration in the Los Angeles basin as a function of 9 predictor
variables.
Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>term</th>
<th>edf</th>
<th>Rank</th>
<th>F</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(vdht)</td>
<td>1.000</td>
<td>1</td>
<td>9.360</td>
<td>0.00242 **</td>
</tr>
<tr>
<td>s(wind)</td>
<td>1.000</td>
<td>1</td>
<td>4.375</td>
<td>0.03729 *</td>
</tr>
<tr>
<td>s(humidity)</td>
<td>3.631</td>
<td>8</td>
<td>2.595</td>
<td>0.00933 **</td>
</tr>
<tr>
<td>s(temp)</td>
<td>4.361</td>
<td>9</td>
<td>4.694</td>
<td>7.56e-06 ***</td>
</tr>
<tr>
<td>s(ibht)</td>
<td>3.043</td>
<td>7</td>
<td>1.708</td>
<td>0.10658</td>
</tr>
<tr>
<td>s(dgpg)</td>
<td>3.230</td>
<td>7</td>
<td>7.916</td>
<td>7.94e-09 ***</td>
</tr>
<tr>
<td>s(ibtp)</td>
<td>1.939</td>
<td>4</td>
<td>1.809</td>
<td>0.12698</td>
</tr>
<tr>
<td>s(vsty)</td>
<td>2.232</td>
<td>5</td>
<td>3.825</td>
<td>0.00225 **</td>
</tr>
<tr>
<td>s(day)</td>
<td>4.021</td>
<td>9</td>
<td>10.174</td>
<td>1.04e-13 ***</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) = 0.797  Deviance explained = 81.2%
GCV score = 14.137  Scale est. = 13.046  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 3.2 or from the P-values in the summary output from R.

The fit of the model can be checked via residual analysis as in chapter 0. The Tukey-Anscombe plot indicates heteroscedastic errors. We thus try the log-transform for the response variable and re-do the whole analysis. Also, we omit observation (no. 72) which is an outlier wrt. wind speed. The results are given below. Further data analysis would now also consider dropping non-significant variables, possible two-way interactions, etc. This is one motivation for the following section.

Figure 3.2: Estimated function \( \hat{g}_j \) for the ozone data set.
Figure 3.3: Tukey-Anscombe plot for the ozone data set.

Figure 3.4: Estimated additive functions and Tukey-Anscombe plot for the ozone dataset with $\log(O_3)$ as response variable and after omitting one outlier.
3.3 MARS

MARS is a shortcut for multivariate adaptive regression splines. 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

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

and \((d - x_j)_+\). The value \(d\) is a knot and \(x_j\) \((j \in \{1, \ldots, p\})\) is the \(j\)-th component of \(x \in \mathbb{R}^p\). The pair \((x_j - d)_+, (d - x_j)_+\) is called a reflected pair. Note that we use \((x_j - d)_+\) as a function of \(x \in \mathbb{R}^p\) where only the \(j\)th component of \(x\) is relevant. The collection of reflected pairs is then a set of basis functions

\[B = \{(x_j - d)_+, (d - x_j)_+; \quad d \in \{x_{1,j}, x_{2,j}, \ldots, x_{n,j}\}, \quad j \in \{1, \ldots, p\}\},\]

where we note that in the univariate case \((p = 1)\), \(B\) spans the space of continuous piecewise linear functions, i.e., linear splines.

MARS employs now a forward selection of reflected pairs of basis functions in \(B\) 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 \(B\) or a product of functions from \(B\). The model building technique is going in a forward way as follows:

1. Start with the function \(h_0(x) \equiv 1\). Initialize the model set \(M = \{h_0(\cdot) = 1\}\). 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 best pair of functions \((h_{2r-1}(\cdot), h_{2r}(\cdot))\) which are of the form
   \begin{align*}
   h_{2r-1}(\cdot) &= h_\ell(\cdot) \times (x_j - d)_+, \\
   h_{2r}(\cdot) &= h_\ell(\cdot) \times (d - x_j)_+, \quad (3.1)
   \end{align*}

   for some \(h_\ell\) in the model set \(M\) which does not contain \(x_j\)\(^3\) and some basis functions in \(B\). The best pair of functions is defined to be the one which reduces residual sum of squares most. The model fit is then

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

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

   \[M = M_{\text{old}} \cup \{h_{2r-1}(\cdot), h_{2r}(\cdot)\},\]

   with the functions \(h_{2r-1}, h_{2r}\) from (3.1).

\(^3\)consequently, each \(h_r(\cdot)\) will be a product of basis functions (in \(B\)) but contain a predictor variable \(x_j\) at most once.
3. Iterate step 2 until a large enough number of basis functions $h_m(\cdot)$ has been fitted.

4. Do backward deletion ("pruning"), allowing to remove single functions from a pair $h_{2r-1}(\cdot), h_{2r}(\cdot)$; i.e., of all single basis functions, delete the one which increases 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:

$$h_0(x) = 1, \quad M = \{1\},$$
$$h_1(x) = (x_2 - x_7^2)_+, \quad h_2(x) = (x_7^2 - x_2)_+, \quad M = \{1, (x_2 - x_7^2)_+, (x_7^2 - x_2)_+\},$$
$$h_3(x) = (x_7^2 - x_2)_+ \cdot (x_1 - x_5^1)_+, \quad h_4(x) = (x_7^2 - x_2)_+ \cdot (x_5^1 - x_1)_+,$$
$$M = \{1, (x_2 - x_7^2)_+, (x_7^2 - x_2)_+, (x_7^2 - x_2)_+ \cdot (x_1 - x_5^1)_+, h_4(x) \}$$

### 3.3.1 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.

### 3.3.2 MARS in R

MARS has been implemented in S and R in the function `mars` from the package `mda`. A new, slightly more flexible alternative implementation is in package `earth` with function `earth` and syntax

```r
> fit <- earth(Volume ~ ., data = trees)
> summary(fit)

Expression:
  23.20824 + 5.745962 * pmax(0, Girth - 12.9)
- 2.866452 * pmax(0, 12.9 - Girth)
+ 0.7183364 * pmax(0, Height - 76)

Number of cases: 31
Selected 4 of 5 terms, and 2 of 2 predictors
Number of terms at each degree of interaction: 1 3 (additive model)
GCV: 11.48697  RSS: 213.4354  GRSq: 0.958869  RSq: 0.9736697

> predict(fit, data.frame(Girth= 5:15, Height= seq(60,80, length=11)))
```

.........
3.4 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 focus here on feedforward neural nets with one hidden layer. The model is

\[ g_k(x) = f_0 \left( \alpha_k + \sum_{h=1}^{q} w_{hk} \phi \left( \tilde{\alpha}_h + \sum_{j=1}^{p} \tilde{w}_{jh} x_j \right) \right). \]  

(3.2)

which is predicting multivariate \((g_k)_k\) where for regression only \(g_0\) is used and for classification, one typically uses \(g_0, \ldots, g_{J-1}\) and \(\hat{C}(x) := \arg \max_j \hat{g}_j(x)\) which is called “softmax” in the NN literature. The function \(\phi(\cdot)\) is usually the sigmoid function

\[ \phi(x) = \frac{\exp(x)}{1 + \exp(x)}, \]

whereas \(f_0(\cdot)\) is typically chosen as the identity for regression and as the sigmoid \(\phi(x)\) for softmax classification. The \(w_{hk}, \tilde{w}_{jh}, \alpha_k, \tilde{\alpha}_h\) all are unknown parameters. The so-called

![Figure 3.5: Simple “feedforward” neural net with one hidden layer.](image)

input layer consists of the \(p\) predictor variables; the values \(w_{hk} \phi \left( \tilde{\alpha}_h + \sum_{j=1}^{p} \tilde{w}_{jh} x_j \right) \) \((h = 1, \ldots, q)\) make the so-called hidden layer with \(q\) units. And the so-called output layer is just a single unit for univariate regression.

A useful variant of (3.2) is a model which includes a linear regression component:

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

(3.3)

3.4.1 Fitting neural networks in R

Feedforward neural nets can be fitted in R with the function \texttt{nnet} from the package \texttt{nnet}; the algorithm is numerically maximizing the likelihood and basically equivalent to using \texttt{optim(*, method="BFGS")}. 
3.4 Neural Networks

In practice, it can be very important to center and scale all the predictor variables so that they are approximately on the same scale (about “1”). This avoids that gradient methods for optimizing the likelihood get stuck in the “flat regions” of the sigmoid functions. Further note that one can regularize the problem using so called weight decay which stabilizes the algorithm (less dependence on random starting values \( w_{*,*} \)) and diminishes the importance of choosing the number of hidden units \( q \).

library(nnet)
## Using 'd.ozone' data from above, using \( \log(03) \)
## and scaling the x-variables (to mean = 0, sd = 1):
sc.ozone <- data.frame(scale(d.ozone)[, -1],
                      log.O3 = log(d.ozone[, "ozone"]))

> set.seed(22) ## (also try others; nnet() uses random starting values!)
> fit <- nnet(log.O3 ~ ., data = sc.ozone, size = 3, skip = TRUE,
+     decay = 4e-4, linout = TRUE, maxit = 500)
# weights: 43
initial value 294.3043
iter 10 value 568.3197
......
final value 34.631788
converged
> sum(residuals(fit)^2) # -> 34.211
> summary(fit)
a 9-3-1 network with 43 weights
options were - skip-layer connections linear output units decay=4e-04
b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
-2.28 1.27 -0.34 -2.57 1.46 0.03 0.10 -1.02 -0.39 -0.33
b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2
-12.43 5.08 2.04 8.19 -7.66 -7.01 2.40 -0.31 3.58 -1.19
b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3 i9->h3
-19.79 -6.65 1.49 -4.53 -3.95 2.28 6.06 5.19 10.06 -0.20
b->o h1->o h2->o h3->o i1->o i2->o i3->o i4->o i5->o i6->o
2.50 -1.81 0.68 0.71 0.11 -0.09 -0.49 0.72 0.01 -0.03
i7->o i8->o i9->o
0.03 -0.29 -0.15

## without linear model component: skip=FALSE
set.seed(22)
> fit1 <- nnet(log.O3 ~ ., data = sc.ozone, size = 3, skip = FALSE,
+     decay = 4e-4, linout = TRUE, maxit = 500)
..............
final value 42.909
converged
> sum(residuals(fit1)^2) # 41.865
> summary(fit1)
a 9-3-1 network with 34 weights
options were - linear output units decay=4e-04
b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1 i9->h1
-27.22 8.91 0.51 8.78 -8.26 10.55 -8.46 11.89 -11.51 8.51
b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2 i9->h2
linout=TRUE indicates that the function is fitted on the linear regression scale; for classification, we use the default linout=FALSE. size = 3 chooses 3 hidden units, regularized by weight decay = 4e-4: these are tuning parameters, and skip=TRUE enforces a neural net with a linear model component. The random seed is stored because, by default, nnet() uses random starting values for the high-dimensional optimization.

3.5 Projection pursuit regression

Projection pursuit regression (for regression problems) bears some similarities to feedforward neural networks. Instead of taking a linear combination of $q$ different sigmoid function outputs (from the $q$ hidden units in the hidden layer) in (3.2), we use the following model:

$$g_{PPR}(x) = \mu + \sum_{k=1}^{q} f_k(\sum_{j=1}^{p} \alpha_{jk} x_j), \quad \text{where}$$

$$\sum_{j=1}^{p} \alpha_{jk}^2 = 1, \quad \mathbb{E}[f_k(\sum_{j=1}^{p} \alpha_{jk} x_j)] = 0, \quad \text{for all } k.$$

The functions $f_k() : \mathbb{R} \rightarrow \mathbb{R}$ are nonparametric (i.e. “arbitrary” smooth); the linear combinations $\sum_{j=1}^{p} \alpha_{jk} x_j$ are linear projections: For the unit vector $\alpha_k = (\alpha_{1k}, \ldots, \alpha_{pk})^\top$, $\alpha_k^\top x$ is the projection of $x$ onto (a ray through $\vec{0}$ in direction) $\alpha_k$. Note that the function $x \mapsto f_k(\sum_{j=1}^{p} \alpha_{jk} x_j)$ only varies along the direction $\alpha_k$, hence the $f_k$’s are called ridge functions and the model “projection pursuit”. This model typically requires much smaller $q$ than the hidden units 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).

3.5.1 Projection pursuit regression in R

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

$$g_{PPR}(x) = \mu + \sum_{k=1}^{q} \beta_k f_k(\sum_{j=1}^{p} \alpha_{jk} x_j),$$

$$\sum_{j=1}^{p} \alpha_{jk}^2 = 1, \quad \mathbb{E}[f_k(\sum_{j=1}^{p} \alpha_{jk} x_j)] = 0, \quad \text{Var}(f_k(\sum_{j=1}^{p} \alpha_{jk} x_j)) = 1, \quad \text{for all } k.$$

Consider the ozone data set from above.

```r
fit <- ppr(log(O3) ~ . , data = d.ozone, nterms=4)  # nterms specifies the number of ridge functions
sfsmisc::mult.fig(4)  # or just par(mfrow=c(2,2))
```
plot(fit) ## 4 terms -> 4 plots

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

The estimated ridge functions are shown in Figure 3.6.

Figure 3.6: Four estimated ridge functions $f_k(\cdot)$ from a projection pursuit fit to the ozone data ($\log(O_3)$ ~ .).