Chapter 9: Additive Models[a]

In this chapter we consider data points $\{(y_i^0, x_i^0)\}_{i=1}^n$ in $\mathbb{R} \times \mathbb{R}^p$, and assume the following regression model

$$Y_i^0 = f(x_i^0) + \epsilon_i, \qquad i = 1, \dots n \tag{9.1}$$

where $\mathbb{E}[\epsilon_i] = 0$, $\mathbb{E}[\epsilon_i^2] = \sigma^2$ and $\mathbb{E}[\epsilon_i \epsilon_l] = 0$ for all $i \neq l$ and where $f \in \mathcal{F} \subseteq \mathcal{C}^2(\mathbb{R}^p)$ with $\mathcal{F}$ an infinite dimensional space.

Ideally, we want the class of functions $\mathcal{F}$ to be large to make the model (9.1) very flexible or, equivalently, to impose little assumptions on f. However, two problems arise if the set $\mathcal{F}$ is too rich:

- Computing the estimated function $\hat{f}$ may be computationally expensive (as this is the case e.g. for multi-dimensional smoothing models, which assume $\mathcal{F} = \mathcal{C}^2(\mathbb{R}^p)$; see Chapter 8).

- The convergence rate of the estimated function (towards some limiting function $f_0 \in \mathcal{F}$, assuming that such a function exists) is typically of size $\mathcal{O}(n^{-\alpha_p})$, where $\alpha_p \in (0, 1/2]$ decreases as p increases (see Chapter 11 for some formal results).

To avoid these two potential problems, in additive models the set $\mathcal{F}$ is defined by

$$\mathcal{F} = \Big\{ f \in \mathcal{C}^2(\mathbb{R}^p) \text{ s.t. } f(z) = \sum_{j=1}^p f_j(z_j), \, \forall z \in \mathbb{R}^p, \\ \text{where } f_j \in \mathcal{C}^2(\mathbb{R}) \text{ for all } j \in \{1, \dots, p\} \Big\} \tag{9.2}$$

so that only functions of a single variable need to be estimated.

[a] The main references for this chapter is [3, Section 9.1.1] and [14, Section 4.3].

Estimation of f in the model (9.1)-(9.2)

Following the smoothing approach discussed in Chapter 8, the considered estimator $\{\tilde{f}_{\lambda,j}\}_{j=1}^p$ of $\{f_j\}_{j=1}^p$ is such that

$$\{\tilde{f}_{\lambda,j}\}_{j=1}^p \in \underset{f_1,\ldots,f_p\in\mathcal{C}^2(\mathbb{R})}{\operatorname{argmin}} \sum_{i=1}^n \Big(y_i^0 - \sum_{j=1}^p f_j(x_{ij}^0)\Big)^2 + \sum_{j=1}^p \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x \tag{9.3}$$

where the p penalty terms play the same role as in smoothing; i.e. the jth penalty term controls the wiggliness of the estimated function $\hat{f}_{\lambda,j}$.

For very j we let $x_{(j),\min}^0 = \min\{x_{ij}^0\}_{i=1}^n$, $x_{(j),\max}^0 = \max\{x_{ij}^0\}_{i=1}^n$, $\tilde{x}_{(j)}^0$ denote the set of distinct elements of $\{x_{ij}^0\}_{i=1}^n$ and $m_j = |\tilde{x}_{(j)}^0|$.

For $p = 1$ we remark that (9.3) reduces to the smoothing problem (8.2), in which case, by Theorem 8.2, $\tilde{f}_{\lambda,1} : [x_{(1),\min}^0, x_{(1),\max}^0] \to \mathbb{R}$ is a natural cubic spline with knots $\tilde{x}_{(1)}^0$ and $\tilde{f}_{\lambda,1}''(x) = 0$ for all $x \notin [x_{(1),\min}^0, x_{(1),\max}^0]$.

The following result extends the conclusion of Theorem 8.2 to any $p \geq 1$.

Theorem 9.1 *Let $\{\tilde{f}_{\lambda,j}\}_{j=1}^p$ be as in (9.3). Then, for all $j \in \{1,\ldots,p\}$ the function $\tilde{f}_{\lambda,j} : [x_{(j),\min}^0, x_{(j),\max}^0] \to \mathbb{R}$ belongs to the set $\mathcal{S}_{m_j}^*(\tilde{x}_{(j)}^0)$ and $\tilde{f}_{\lambda,j}''(x) = 0$ for all $x \notin [x_{(j),\min}^0, x_{(j),\max}^0]$.*

Proof: Theorem 9.1 follows from similar computations as the ones used to prove Theorem 8.2. □.

Identifiability issues

For all $p \geq 2$ the functions $\{\tilde{f}_{\lambda,j}\}_{j=1}^p$ are not uniquely defined since, for instance, for all $c \in \mathbb{R}$ we have

$$Y_i^0 = \sum_{j=1}^p f_j(x_{ij}^0) + \epsilon_i = \big(f_1(x_{i1}^0) - c\big) + \big(f_2(x_{i2}^0) + c\big) + \sum_{j=3}^p f_j(x_{ij}^0) + \epsilon_i$$

where $f_1 - c$ and $f_2 + c$ are in $\mathcal{C}^2(\mathbb{R})$ if $f_1, f_2 \in \mathcal{C}^2(\mathbb{R})$.

The standard approach to solve this identification problem is to add an intercept α to the model and to impose that $\sum_{i=1}^n f_j(x_{ij}^0) = 0$ for all j, that is, instead of considering (9.1) to consider the model

$$Y_i^0 = \alpha + \sum_{j=1}^p f_j(x_{ij}^0) + \epsilon_i, \quad i = 1, \ldots, n \tag{9.4}$$

where $f_j \in \mathcal{C}_j^2(\mathbb{R}) := \{f \in \mathcal{C}^2(\mathbb{R}) : \sum_{i=1}^n f(x_{ij}^0) = 0\}$ for all j.

Then, letting $\tilde{\mathcal{C}}(\mathbb{R}) = \times_{j=1}^p \mathcal{C}_j(\mathbb{R})$, we estimate α and $\{f_j\}_{j=1}^p$ using

$$\big(\hat{\alpha}, \{\hat{f}_{\lambda,j}\}_{j=1}^p\big)$$
$$\in \underset{\alpha\in\mathbb{R},\, f\in\tilde{\mathcal{C}}^2(\mathbb{R})}{\operatorname{argmin}} \sum_{i=1}^n \Big(y_i^0 - \alpha - \sum_{j=1}^p f_j(x_{ij}^0)\Big)^2 + \sum_{j=1}^p \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x.$$

It is direct to see that $\hat{\alpha} = \frac{1}{n}\sum_{i=1}^n y_i^0$, and thus that

$$\{\hat{f}_{\lambda,j}\}_{j=1}^p \in \underset{f\in\tilde{\mathcal{C}}^2(\mathbb{R})}{\operatorname{argmin}} \sum_{i=1}^n \Big(y_i - \sum_{j=1}^p f_j(x_{ij}^0)\Big)^2 + \sum_{j=1}^p \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x. \tag{9.5}$$

It can be shown that if the matrix $\boldsymbol{X}^0$ has full rank column then the functions $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ are uniquely defined, while if $\operatorname{rank}(\boldsymbol{X}^0) < p$ only the non-linear parts of the functions $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ are uniquely defined [4, Section 9.1.1, page 297].

Computation of $\{\hat{f}_{\lambda,j}\}_{j=1}^p$

The following result shows that the conclusion of Theorem 9.1 for $\{\tilde{f}_{\lambda,j}\}_{j=1}^p$ satisfying (9.3) also holds for $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ satisfying (9.5).

Theorem 9.2 *Let $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ be as in (9.5). Then, for all $j \in \{1,\ldots,p\}$ the function $\hat{f}_{\lambda,j} : [x^0_{(j),\min}, x^0_{(j),\max}] \to \mathbb{R}$ belongs to the set $\mathcal{S}^*_{m_j}(\tilde{x}^0_{(j)})$ and $\hat{f}''_{\lambda,j}(x) = 0$ for all $x \notin [x^0_{(j),\min}, x^0_{(j),\max}]$.*

Proof: Theorem 9.2 follows from similar computations as the ones used to prove Theorem 8.2. □.

For all j we now let $\{b_{(j),k}\}_{k=1}^{m_j}$ be a basis of the set $\mathcal{S}^*_{m_i}(\tilde{x}^0_j)$ so that, by Theorem 9.2, there exists a $\beta_{(j),\lambda} \in \mathbb{R}^{m_j}$ such that

$$\hat{f}_{\lambda,j}(x) = \sum_{k=1}^{m_j} \beta_{(j),\lambda,k}\, b_{(j),k}(x), \quad \forall x \in [x^0_{(j),\min}, x^0_{(j),\max}] \tag{9.6}$$

and therefore the problem of computing $\hat{f}_{\lambda,j}$ reduces to the problem of computing $\beta_{(j),\lambda}$.

Let $m = \sum_{j=1}^p m_j$, $\beta = (\beta_{(1)},\ldots,\beta_{(p)}) \in \mathbb{R}^m$ with $\beta_{(j)} \in \mathbb{R}^{m_j}$, $\boldsymbol{Z} = [\boldsymbol{Z}_{(1)}\ldots\boldsymbol{Z}_{(p)}]$ with $\boldsymbol{Z}_{(j)}$ is the $n \times m_j$ matrix having $b_{(j),k}(x^0_{ij})$ as element (i,k), $\boldsymbol{S}_\lambda = \text{diag}(\lambda_1 \boldsymbol{S}_{(1),\text{pen}},\ldots,\lambda_p \boldsymbol{S}_{(p),\text{pen}})$ where $\boldsymbol{S}_{(j),\text{pen}}$ is the $m_j \times m_j$ matrix having $\int_{[x^0_{(j),\min}, x^0_{(j),\max}]} b''_{(j),k}(x) b''_{(j),l}(x)\text{d}x$ as element (k,l), and let $\mathcal{B}_j = \{\beta \in \mathbb{R}^{m_j} : \sum_{i=1}^n \sum_{k=1}^{m_j} \beta_{(j),k} b_{(j),k}(x^0_{ij}) = 0\}$.

Remark: The set $\mathcal{B} := \times_{j=1}^p \mathcal{B}_j$ is the set of all $\beta \in \mathbb{R}^m$ such that

$$\sum_{i=1}^n \sum_{k=1}^{m_j} \beta_{(j),k} b_{(j),k}(x^0_{ij}) = 0 \quad \forall j \in \{1,\ldots,p\}.$$

Computation of β_λ

Using the above notation, it follows that $\beta_\lambda := (\beta_{(1),\lambda}, \ldots, \beta_{(p),\lambda})$ is such that (9.6) holds if and only if

$$\beta_\lambda \in \operatorname*{argmin}_{\beta \in \mathcal{B}} F_\lambda(\beta), \quad F_\lambda(\beta) = \|y - \boldsymbol{Z}\beta\|^2 + \beta^\top \boldsymbol{S}_\lambda \beta. \tag{9.7}$$

A convenient approach to solve the constraint, and typically high-dimensional, optimization problem (9.7) is to use a coordinate descent algorithm (see Chapter 7, page 133) where each sub-vector $\beta_{(j)}$ of β is treated as a coordinate.

To this aim, for $\tilde{\beta} \in \mathcal{B}$ we need to compute (with obvious notation and obvious convention when $j \in \{1, p\}$),

$$\begin{aligned}
\beta_{(j)}^{(\tilde{\beta})} &\in \operatorname*{argmin}_{\beta_{(j)} \in \mathcal{B}_j} F_\lambda\big(\tilde{\beta}_{(1)}, \ldots, \tilde{\beta}_{(j-1)}, \beta_{(j)}, \tilde{\beta}_{(j+1)}, \ldots, \tilde{\beta}_{(p)}\big) \\
&= \operatorname*{argmin}_{\beta_{(j)} \in \mathcal{B}_j} \sum_{i=1}^n \Big(\tilde{y}_i - \sum_{k=1}^{m_j} \beta_{(j),k} b_{(j),k}(x_{ij}^0)\Big)^2 + \lambda_j \beta_{(j)}^\top \boldsymbol{S}_{(j),\text{pen}} \beta_{(j)}
\end{aligned}$$

where $\tilde{y}_i = y_i - \sum_{l \neq j} \sum_{k=1}^{m_l} \tilde{\beta}_{(l),k} b_{(l),k}(x_{il}^0)$ for all $i \in \{1, \ldots, n\}$.

Remark now that $\sum_{i=1}^n \tilde{y}_i = 0$, and thus

$$\begin{aligned}
&\inf_{\beta_{(j)} \in \mathcal{B}_j} \sum_{i=1}^n \Big(\tilde{y}_i - \sum_{k=1}^{m_j} \beta_{(j),k} b_{(j),k}(x_{ij}^0)\Big)^2 + \lambda_j \beta_{(j)}^\top \boldsymbol{S}_{(j),\text{pen}} \beta_{(j)} \\
&= \inf_{f_j \in \mathcal{C}_j^2(\mathbb{R})} \sum_{i=1}^n \Big(\tilde{y}_i - f_j(x_{ij}^0)\Big)^2 + \lambda_j \int_{\mathbb{R}} (f_j''(x))^2 \mathrm{d}x \qquad (9.8) \\
&= \inf_{f_j \in \mathcal{C}^2(\mathbb{R})} \sum_{i=1}^n \big(\tilde{y}_i - f_j(x_{ij}^0)\big)^2 + \lambda_j \int_{\mathbb{R}} (f_j''(x))^2 \mathrm{d}x
\end{aligned}$$

where the first equality holds by applying Proposition 8.3 with $w_i^0 = 0$ for all $i \in \{1, \ldots, n\}$.

A coordinate descent algorithm for computing β_λ

Therefore, by (9.8), and using Theorem 8.2 and Corollary 8.1,we have

$$\begin{aligned}\beta_{(j)}^{(\tilde{\beta})} &\in \underset{\beta_{(j)}\in\mathbb{R}^{m_j}}{\operatorname{argmin}} \sum_{i=1}^{n}\Big(\tilde{y}_i - \sum_{k=1}^{m_j}\beta_{(j),k}b_{(j),k}(x_{ij}^0)\Big)^2 + \lambda_j\big(\beta^{(j)}\big)^\top \boldsymbol{S}_{(j),\text{pen}}\beta^{(j)} \\ &= \big(\boldsymbol{Z}_{(j)}^\top\boldsymbol{Z}_{(j)} + \lambda_j\boldsymbol{S}_{(j),\text{pen}}\big)^{-1}\boldsymbol{Z}_{(j)}^\top\tilde{y}\end{aligned}$$

leading to the following coordinate descent algorithm for computing β_λ.

Coordinate descent algorithm for computing β_λ.

Let $\beta^0 = 0 \in \mathbb{R}^m$
for $k \geq 1$ **do**
  for $j = 1, \ldots, p$ **do**
    (i) With the convention that empty sums are null let

$$\tilde{y}_i^{(k,j)} = y_i - \sum_{l=1}^{j-1}\sum_{q=1}^{m_l}\beta_{(l),q}^{(k)}b_{(l),k}(x_{il}^0) - \sum_{l=j+1}^{p}\sum_{q=1}^{m_l}\beta_{(l),q}^{(k-1)}b_{(l),k}(x_{il}^0), \quad \forall i.$$

    (ii) With obvious convention when $j \in \{1, p\}$ let

$$\beta_{(j)}^{(k)} = \big(\boldsymbol{Z}_{(j)}^\top\boldsymbol{Z}_{(j)} + \lambda_j\boldsymbol{S}_{(j),\text{pen}}\big)^{-1}\boldsymbol{Z}_{(j)}^\top\tilde{y}^{(k,j)}$$

  end for
  (ii): Set $\beta^{(k)} = (\beta_{(1)}^{(k)}, \ldots, \beta_{(p)}^{(k)})$
  if Convergence=TRUE **then**
    return $\beta^{(k)}$ and **break**
  end if
end for

Remark: Letting $\beta^0 = 0$ ensures that $\sum_{i=1}^n \tilde{y}_i^{(k,j)} = 0$ for all k and j.

The backfitting algorithm for computing $\{\hat{f}_{\lambda,j}\}_{j=1}^p$

The above coordinate descent algorithm for computing β_λ is known as the backfitting algorithm, and is usually presented in the following equivalent way.

The backfitting algorithm for computing $\{\hat{f}_{\lambda,j}\}_{j=1}^p$

Let $\hat{f}_{\lambda,j} \equiv 0$ for all $j \in \{1, \dots, p\}$.
for $k \geq 1$ **do**
  for $j = 1, \dots, p$ **do**
    (i) Let $\tilde{y}_i^{(k,j)} = y_i - \sum_{l \neq j}^p \hat{f}_{\lambda,l}(x_{il}^0)$ for $i = 1, \dots, n$
    (ii) Let

$$f_{\lambda,j} = \underset{f_j \in \mathcal{C}^2(\mathbb{R})}{\operatorname{argmin}} \sum_{i=1}^n \left(\tilde{y}_i^{(k,j)} - f_j(x_{ij}^0)\right)^2 + \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x$$

    (iii) Let $\hat{f}_{\lambda,j} = f_{\lambda,j} - \frac{1}{n}\sum_{i=1}^n f_{\lambda,j}(x_{ij}^0)$
  end for
  if Convergence=TRUE **then**
    return $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ and **break**
  end if
end for

Remark: Step (ii) of the algorithm is a one-dimensional smoothing problem and can be solved in $\mathcal{O}(n \log n)$ operations (see Chapter 8).

Remark: In theory step (iii) can be omitted since, as $\sum_{i=1}^n \tilde{y}_i^{(k,j)} = 0$, we have $\sum_{i=1}^n f_{\lambda,j}(x_{ij}^0) = 0$. However, in practice, this latter equality may not hold exactly due to rounding errors.

Remark: It can be shown that the backfitting algorithm always converges, even when $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ is not unique.

Limitations of the backfitting approach for estimating f

The backfitting algorithm has the nice property to be a valid algorithm for computing an optimal solution $\{\hat{f}_{\lambda,j}\}_{j=1}^{p}$ of the optimization problem (9.5). However, with this approach for estimating f in the model (9.1)-(9.2), choosing λ is a challenging task.

To understand this latter point assume that we want to choose λ by minimizing the leave-one-out cross-validation criterion

$$\mathrm{OCV}_{\mathrm{add}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}\hat{f}_{\lambda,j}^{(-i)}(x_{ij}^{0})\right)^2$$

where $\{\hat{f}_{\lambda,j}^{(-i)}\}_{j=1}^{p}$ is the estimate of f in the model (9.1)-(9.2) obtained by the backfitting algorithm after having removed the ith data point.

- Because (9.7) is a constraint optimization problem Theorem 6.1 does not apply and thus computing $\mathrm{OCV}_{\mathrm{add}}(\lambda)$ requires to fit n times the model (9.1)-(9.2) (each time using $n-1$ observations).

- λ is a vector of dimension p and thus even for small values of $p > 1$ it is computationally expensive to find a good approximation of $\hat{\lambda} \in \mathrm{argmin}_{\lambda\in\mathbb{R}^m} \mathrm{OCV}_{\mathrm{add}}(\lambda)$ by computing $\mathrm{OCV}_{\mathrm{add}}(\lambda)$ for all λ in a finite set $\Lambda_p \subset \mathbb{R}^m$, as the size of the set Λ_p needed to have a good approximation of $\hat{\lambda}$ grows exponentially fast with p.

- Expanding on the previous point, we cannot used a gradient based optimization method to minimize the function $\mathrm{OCV}_{\mathrm{add}}(\cdot)$, the main reason being that because (9.7) is a constraint optimization problem it is unclear how to compute $\nabla_\lambda \beta_\lambda$[a].

[a] We could replace $\nabla\mathrm{OCV}_{\mathrm{add}}(\lambda)$ by a numerical derivative but that would be computationally expensive.

Another approach for estimating f: preliminaries

Recall that our objective is to solve (9.5), that is to compute functions $\{\hat{f}_{\lambda,j}\}_{j=1}^p$ such that

$$\{\hat{f}_{\lambda,j}\}_{j=1}^p \in \operatorname*{argmin}_{f\in\tilde{\mathcal{C}}^2(\mathbb{R})} \sum_{i=1}^n \Big(y_i - \sum_{j=1}^p f_j(x_{ij}^0)\Big)^2 + \sum_{j=1}^p \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x.$$

Let $j \in \{1,\dots,p\}$ and let

$$\tilde{b}_{(j),k} = b_{(j),k} - \frac{1}{n}\sum_{i=1}^n b_{(j),k}(x_{ij}^0), \quad \forall k \in \{1,\dots,m_j\}.$$

Then, for every $\beta_{(j)} \in \mathbb{R}^{m_j}$, if $f_j = \sum_{k=1}^{m_j} \beta_{(j),k}\tilde{b}_{(j),k}$ we have

$$\sum_{i=1}^n f_j(x_{ij}^0) = \sum_{i=1}^n \sum_{k=1}^{m_j} \beta_{(j),k}\tilde{b}_{(j),k}(x_{ij}^0) = \sum_{k=1}^{m_j} \beta_{(j),k} \sum_{i=1}^n \tilde{b}_{(j),k}(x_{ij}^0) = 0$$

showing that

$$\operatorname{span}\big(\tilde{b}_{(j),1},\dots,\tilde{b}_{(j),m_j}\big) = \Big\{f_j \in \mathcal{S}_{m_j}^*(\tilde{x}_{(j)}^0) : \sum_{i=1}^n f_j(x_{ij}^0) = 0\Big\}.$$

In words, if for all j we replace in (9.7) the basis functions $\{b_{(j),k}\}_{k=1}^{m_j}$ by the centred basis functions $\{\tilde{b}_{(j),k}\}_{k=1}^{m_j}$ then (9.7) becomes an unconstrained optimization problem.

Another approach for estimating f: Optimization problem

Letting $\tilde{\boldsymbol{Z}} = \boldsymbol{C}_n \boldsymbol{Z}$ it follows that if

$$\beta_\lambda \in \operatorname*{argmin}_{\beta \in \mathbb{R}^m} \|y - \tilde{\boldsymbol{Z}}\beta\|^2 + \beta^\top \boldsymbol{S}_\lambda \beta \tag{9.9}$$

then

$$\left\{ \sum_{k=1}^{m_j} \beta_{\lambda,j} \tilde{b}_{(j),k} \right\}_{j=1}^{p} \in \operatorname*{argmin}_{f \in \tilde{\mathcal{C}}^2(\mathbb{R})} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} f_j(x_{ij}^0) \right)^2 + \sum_{j=1}^{p} \lambda_j \int_{\mathbb{R}} f_j''(x)^2 \mathrm{d}x.$$

Remark: since $\tilde{b}''_{(j),k} = b''_{(j),k}$ for all j and k the penalty matrix $\boldsymbol{S}_\lambda$ in (9.9) is the same as in (9.7).

It is important to note that, unlike the matrix $\boldsymbol{Z}^\top \boldsymbol{Z}$, the matrix $\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}}$ is not full rank when $m \leq n$. Indeed, the original basis functions $\{b_{(j),k}\}_{k=1}^{m_j}$ are such that there exists a $c \in \mathbb{R}^{m_j}$ such that $1 = \sum_{k=1}^{m_j} c_k b_{(j),k}(x)$ for all $x \in [x^0_{(j),\min}, x^0_{(j),\max}]$[a]. Therefore,

$$1 = \sum_{k=1}^{m_j} c_k b_{(j),k}(x_{ij}^0) \quad \forall i \in \{1, \ldots, n\}$$

implying that, for all i,

$$\sum_{k=1}^{m_j} c_k \tilde{b}_{(j),k}(x_{ij}^0) = \sum_{k=1}^{m_j} c_k b_{(j),k}(x_{ij}^0) - \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m_j} c_k b_{(j),k}(x_{ij}^0) = 1 - 1 = 0$$

showing that the columns of the matrix $\tilde{\boldsymbol{Z}}_{(j)} := \boldsymbol{C}_n \boldsymbol{Z}_{(j)}$ are linearly depend. Noting that $\tilde{\boldsymbol{Z}} = [\tilde{\boldsymbol{Z}}_{(1)} \ldots \tilde{\boldsymbol{Z}}_{(p)}]$, it follows that $\operatorname{rank}(\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}}) = \operatorname{rank}(\boldsymbol{Z}^\top \boldsymbol{Z}) - p$.

Remark: The matrix $\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}}$ being not full rank, the matrix $\tilde{\boldsymbol{Z}}^\top \tilde{\boldsymbol{Z}} + \boldsymbol{S}_\lambda$ may not be invertible[b] and thus the solution to (9.9) may not be unique.

[a] See for instance the remark on page 154.

[b] Recall that the matrix $\boldsymbol{S}_\lambda$ is not full rank.

Another approach for estimating f: The procedure

To facilitate the estimation of f, in practice we often proceed as follows:

1. We choose integers $\{m'_j\}_{j=1}^p$ such that $m' := \sum_{j=1}^p m'_j \leq n$ and such that $1 < m'_j \leq m_j$ for all j.
2. For all j, we choose m'_j (evenly spaced) elements $\{\check{x}_{(j),k}\}_{k=1}^{m'_j}$ of $\tilde{x}^0_{(j)}$ (thinning, see Chapter 8).
3. For all j, we choose a basis $\{b_{(j),k}\}_{k=1}^{m'_j}$ of $\mathcal{S}^*_{m'_j}(\{\check{x}_{(j),k}\}_{k=1}^{m'_j})$ and let $\tilde{b}_{(j),k} = b_{(j),k} - \frac{1}{n}\sum_{i=1}^n b_{(j),k}(x^0_{ij})$ for all k.
4. For all j, we form the $n \times m'_j$ matrix $\tilde{\boldsymbol{Z}}_{(j)}$ having $\tilde{b}_{(j),k}(x^0_{ij})$ as element (i,k).
5. For all j, we remove one column of the matrix $\tilde{\boldsymbol{Z}}_{(j)}$ (usually the one having the lowest variance) and the associated row and column of $\boldsymbol{S}_\lambda$. Denote by $\check{\boldsymbol{Z}}_{(j)}$ the resulting $n \times (m'_j - 1)$ matrix and by $\check{\boldsymbol{S}}_\lambda$ the new penalty matrix of the model.
6. For all j we estimate f_j by

$$\check{f}_{\lambda,j} = \sum_{k=1}^{m'_j} \check{\beta}_{(j),k}\tilde{b}_{(j),k} \tag{9.10}$$

where, letting $\check{\boldsymbol{Z}} = [\check{\boldsymbol{Z}}_{(1)}\ldots\check{\boldsymbol{Z}}_{(p)}]$,

$$\check{\beta}_\lambda \in \underset{\beta\in\mathbb{R}^{m'-p}}{\operatorname{argmin}} \|y - \check{\boldsymbol{Z}}\beta\|^2 + \beta^\top\check{\boldsymbol{S}}_\lambda\beta = \left(\check{\boldsymbol{Z}}^\top\check{\boldsymbol{Z}} + \check{\boldsymbol{S}}_\lambda\right)^{-1}\check{\boldsymbol{Z}}^\top y. \tag{9.11}$$

Remark: The functions $\{\check{f}_{\lambda,j}\}_{j=1}^p$ defined in (9.10) provide only an approximate solution to (9.5).

Remark: When m' is large we can compute an approximate solution to (9.11) using the backfilling algorithm introduced earlier in this chapter (where for all j the matrix $\boldsymbol{Z}_{(j)}$ is replaced y the matrix $\check{\boldsymbol{Z}}_{(j)}$).

Another approach for estimating f: Choosing the penalty parameter

The vector $\check{\beta}_\lambda$ satisfies the condition of Theorem 6.1 and therefore we can evaluate the ordinary and generalized cross-validation criteria from a single estimation of the additive model.

In practice, generalized cross-validation is usually preferred, in which case we let $\lambda = \hat{\lambda}$ where

$$\hat{\lambda} \in \operatorname*{argmin}_{\lambda \in \mathbb{R}^p} \mathrm{GCV}_{\mathrm{add}}(\lambda), \quad \mathrm{GCV}_{\mathrm{add}}(\lambda) = \frac{n\|y - \check{\boldsymbol{Z}}\check{\beta}_\lambda\|^2}{(n - \mathrm{tr}(\check{\boldsymbol{A}}_\lambda))^2}$$

where $\check{\boldsymbol{A}}_\lambda = \check{\boldsymbol{Z}}\big(\check{\boldsymbol{Z}}^\top\check{\boldsymbol{Z}} + \check{\boldsymbol{S}}_\lambda\big)^{-1}\check{\boldsymbol{Z}}^\top$.

As mentioned above, even for small values of $p > 1$ we cannot approximate well $\hat{\lambda}$ by evaluating the function $\mathrm{GCV}_{\mathrm{add}}(\cdot)$ for all λ in some finite set Λ.

For this reason, in practice, the function $\mathrm{GCV}_{\mathrm{add}}(\cdot)$ is numerically minimized using Newton's method (see [14, Section 6.5.1, page 273] for more details).

Illustrative example: The ozone dataset[a]

The objective of this example is to study, in the Los Angeles Basin in 1976, the link, between atmospheric ozone concentration and $p = 5$ meteorological variables, namely the temperature (variable temp), the inversion base height (ibh) the inversion top temperature (ibt), the humidity (humidity) and the Daggett pressure gradient (dpg). The dataset contains $n = 330$ observations. For all j, the function f_j is estimated using $m'_j = 50$ basis functions while λ is chosen using GCV.

The resulting estimates of the functions $\{f_j\}_{j=1}^p$ are shown Figure 9.1. We observe that in the fitted model the response variable is linear in the variable humidity, and non-linear in the others.

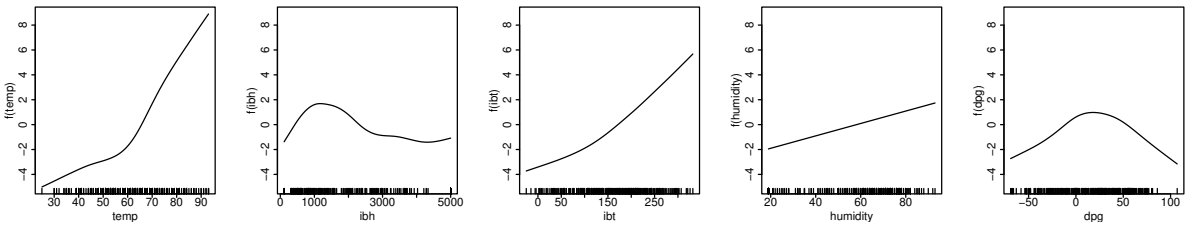

Figure 9.1: Estimation of $\{f_j\}_{j=1}^p$ for the ozone dataset.

To assess further the improvement of the additive model compared to a simple linear regression model, we split the dataset in to a training set of size 247 and a test set of size 83, fit the additive and the linear model on the training set and compare their prediction error on the test. This experiment lead to an average squared prediction error of 6.4 for the additive model, against 19.93 for the linear regression model.

[a]The dataset is available from the `R` package `faraway`.