Chapter 2

Weber Jakob

November 27, 2020

Contents

1 Solution Approach **2**
1.1 Shape-constraint P-splines 3
1.1.1 Monotonic increasing constraint 3
1.1.2 Monotonic decreasing constraint 4
1.1.3 Convex constraint 5
1.1.4 Concave constraint 5
1.1.5 Peak constraint 5
1.1.6 Valley constraint 6
1.1.7 Jamming constraint 6
1.1.8 Boundedness constraint 6
1.2 Univariate Function Approximation 7

Chapter 1

Solution Approach

In this chapter we use the theory discussed in Chapter **??** to estimate uni- and bivariate functions using data and a priori domain knowledge. An overview of the different problems considered in this chapter is given in Table 1.1.

Univariate	**Section**	**Bivariate**	**Section**
B-splines		Tensor-product B-splines	
P-splines		Tensor-product P-splines	
SCP-splines		Tensor-product SCP-splines	

Table 1.1: Problem overview.

First, we are using B-splines, see Section **??**, for the estimation of the unknown function $y = f(x)$, i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_1(\mathbf{y}, \boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|, \tag{1.1}$$

using the B-spline or tensor-product B-spline basis matrix $\mathbf{X}$. Next, we use the concept of P-splines, see Section **??**, to estimate smooth functions, i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_2(\mathbf{y}, \boldsymbol{\beta}; \lambda) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \lambda \cdot \mathrm{pen}(\boldsymbol{\beta}), \tag{1.2}$$

where $\mathrm{pen}(\boldsymbol{\beta})$ specifies a smoothness penalty term. Finally, we are going to incorporate a priori domain knowledge into the fitting process using shape-constrained P-splines (SCP-splines), i.e. we solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}} Q_3(\mathbf{y}, \boldsymbol{\beta}; \lambda, \lambda_c) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \lambda \cdot \mathrm{pen}(\boldsymbol{\beta}) + \lambda_c \cdot \mathrm{con}(\boldsymbol{\beta}), \tag{1.3}$$

where $\mathrm{pen}(\boldsymbol{\beta})$ is again a smoothness penalty term and $\mathrm{con}(\boldsymbol{\beta})$ specifies the user-defined shape-constraint to incorporate a priori domain knowledge with, see [**?**] and [**?**]. Various types a priori domain knowledge can be incorporated using the constraints listed in Table 1.2.

Constraint		Description	Section
Jamming		$f(x^{(M)}) \approx y^{(M)}$	1.1.7
Boundedness	lower	$f(x) \geq M$	1.1.8
	upper	$f(x) \leq M$	1.1.8
Monotonicity	increasing	$f'(x) \geq 0$	1.1.1
	decreasing	$f'(x) \leq 0$	1.1.2
Curvature	convex	$f''(x) \geq 0$	1.1.3
	concave	$f''(x) \leq 0$	1.1.4
Unimodality	peak	$m = \arg\max_x f(x)$ $f'(x) \geq 0 \quad \text{if } x < m$ $f'(x) \leq 0 \quad \text{if } x > m$	1.1.5
	valley	$m = \arg\min_x f(x)$ $f'(x) \leq 0 \quad \text{if } x < m$ $f'(x) \geq 0 \quad \text{if } x > m$	1.1.6

Table 1.2: Overview of the considered constraints

The focus of this chapter is the definition and use of shape-constraint P-splines, which are characterize by their parametes $\boldsymbol{\beta}$ given by solving the optimization problem 1.3.

1.1 Shape-constraint P-splines

In Section **??**, we enforced smoothness by penalizing the second-order derivative of the underlying B-spline using finite differences of adjacent parameters $\boldsymbol{\beta}$ over the whole input space to create the so-called P-splines. We will now utilize the same idea to create the shape-constrained penalty term $\text{con}(\boldsymbol{\beta})$ in 1.3. We motivate the approach using the example of monotonic increasing behavior. The descriptions for the other constraints listed in Table 1.2 follow later.

1.1.1 Monotonic increasing constraint

A function is monotonic increasing if it's first-order derivative is larger than or equal to zero for the whole input space. We therefore introduce a penalty of the form

$$\lambda_c \int \left(f(x)'\right)^2 dx \quad \text{if } f'(x) < 0, \tag{1.4}$$

to penalize negative first-order derivatives of the estimated function. The penalty is weighted by the *constraint parameter* λ_c. Making use of the finite difference approximation, see Section **??**, this time for the first-order derivative, leads to a penalty of the form

$$\lambda_c \cdot \text{con}(\boldsymbol{\beta}) = \lambda_c \boldsymbol{\beta}^{\mathrm{T}} \mathbf{K_c} \boldsymbol{\beta}, \tag{1.5}$$

with the shape-constraint penalty matrix $\mathbf{K}_c = \mathbf{D}_1^{\mathrm{T}} \mathbf{V}_c \mathbf{D}_1 \in \mathbb{R}^{d \times d}$. The first-order derivative is approximated using the matrix form of the first-order finite difference operator $\Delta^1 \beta_j = \beta_j - \beta_{j-1}$ given by

$$\mathbf{D}_1\boldsymbol{\beta} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \tag{1.6}$$

with the mapping matrix $\mathbf{D}_1 \in \mathbb{R}^{(d-1)\times d}$. The weighting matrix $\mathbf{V}_c \in \mathbb{R}^{(d-1)\times(d-1)}$ in 1.5 handles the "if"-condition in 1.4. It is a diagonal matrix with the diagonal elements v_j defined as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1\beta_j \geq 0 \\ 1, & \text{if } \Delta^1\beta_j < 0 \end{cases} \quad \text{for } j = 2, 3, \ldots, d. \tag{1.7}$$

Therefor, the weighting matrix $\mathbf{V} := \mathbf{V}(\boldsymbol{\beta})$ depends on the parameters $\boldsymbol{\beta}$ and we arrive at a formulation similar to Ridge regression with a parameter dependent, non-linear penalty matrix $\mathbf{K} := \mathbf{K}(\boldsymbol{\beta})$, see Section **??**. The convex objective function is then of the form

$$Q_3(\mathbf{y}, \boldsymbol{\beta}; \lambda, \lambda_c) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\| + \lambda\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}\boldsymbol{\beta} + \lambda_c\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_c\boldsymbol{\beta}, \tag{1.8}$$

see Appendix B for the proof of convexity. We use the iterative approach given in Algorithm 1 to estimate the optimal parameters $\hat{\boldsymbol{\beta}}_{SCP}$ under the user-defined shape constraint.

Algorithm 1: Estimation of the shape-constraint P-spline coefficients.6

Result: $\hat{\boldsymbol{\beta}}_{SCP}$
$\hat{\boldsymbol{\beta}}_{init} \leftarrow$ Solution from1.1 or 1.2;
$\mathbf{V}_1 \leftarrow \mathbf{V}_c(\hat{\boldsymbol{\beta}}_{init})$;
$\mathbf{V}_0 \leftarrow \mathbf{0}$;
$i \leftarrow 0$;
while $\mathbf{V}_i \neq \mathbf{V}_{i+1}$ **do**
    $\hat{\boldsymbol{\beta}}_{i+1} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{D}_2^{\mathrm{T}}\mathbf{D}_2) + \lambda_c\mathbf{D}_1^{\mathrm{T}}\mathbf{V}_i\mathbf{D}_1)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$;
    $\mathbf{V}_{i+1} \leftarrow \mathbf{V}_c(\hat{\boldsymbol{\beta}}_i)$;
    $i \leftarrow i + 1$;
end
$\hat{\boldsymbol{\beta}}_{SCP} \leftarrow \hat{\boldsymbol{\beta}}_i$

In Algorithm 1 we use a Newton-Raphson scheme for the estimation of $\hat{\boldsymbol{\beta}}_i$. For more information, see Appendix B and [**?**]. The approach described above incorporates the shape-constraint as soft constraint depending on the constraint parameter λ_c with the limits of no constraint for $\lambda_c \to 0$ and hard constraint for $\lambda_c \to \infty$. Therefor, λ_c should be set by hand reflecting the users confidence in the a priori domain knowledge.

1.1.2 Monotonic decreasing constraint

Monotonic decreasing behavior can be introduced by penalizing positive first-order derivatives. Therefor, we use the matrix form of the first-order finite difference operator given in 1.6 as mapping matrix and define the diagonal elements of the weight matrix $\mathbf{V}$ as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \leq 0 \\ 1, & \text{if } \Delta^1 \beta_j > 0 \end{cases} \quad \text{for } j = 2, 3, \ldots, d. \tag{1.9}$$

1.1.3 Convex constraint

Convex behavior can be introduced by penalizing negative second-order derivatives. Therefor, we use the matrix form of the second-order finite difference operator $\Delta_2 \beta_j = \beta_j - 2\beta_{j-1} + \beta_{j-2}$ given by

$$\mathbf{D}_2 \boldsymbol{\beta} = \begin{bmatrix} 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}, \tag{1.10}$$

as mapping matrix $\mathbf{D}_2 \in \mathbb{R}^{(d-2)\times d}$ and define the diagonal elements of the weighting matrix $\mathbf{V}$ as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^2 \beta_j \geq 0 \\ 1, & \text{if } \Delta^2 \beta_j < 0 \end{cases} \quad \text{for } j = 3, 4, \ldots, d. \tag{1.11}$$

1.1.4 Concave constraint

Convex behavior can be introduced by penalizing negative second-order derivatives. Therefor, we use the matrix form of the second-order finite difference operator, see 1.10, as mapping matrix $\mathbf{D}_2 \in \mathbb{R}^{(d-2)\times d}$ and define the diagonal elements of the weighting matrix $\mathbf{V}$ as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^2 \beta_j \leq 0 \\ 1, & \text{if } \Delta^2 \beta_j > 0 \end{cases} \quad \text{for } j = 3, 4, \ldots, d. \tag{1.12}$$

1.1.5 Peak constraint

Peak behavior can be introduced by penalizing negative first-order derivatives for the increasing part and positive first-order derivatives for the decreasing part. Therefor, we use the matrix for of the first-order finite difference operator, see 1.6, as mapping matrix $\mathbf{D}$. The weighting matrix $\mathbf{V}$ now has a special structure. First, we find the data point $x_{(textmax)}$ corresponding to the peak value in the data, i.e. $\max = \max_i y^{(i)}$, for$i = 1, 2, \ldots, n$. Next, we identify the dominant B-spline basis function $B_p^l(x)$ around $x^{(\max)}$, i.e. the B-spline basis function with the maximal value at $x^{(\max)}$, such that

$$B_p^l(x^{(\max)}) > B_j^l(x^{(\max)}), \quad \text{for } j = 1, 2, \ldots, p-1, p+1, \ldots, d. \tag{1.13}$$

We now use the index p of the dominant B-spline basis function in the definition of the diagonal elements of the weighting matrix $\mathbf{V}$ as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \geq 0 \\ 1, & \text{if } \Delta^1 \beta_j < 0 \end{cases}, \quad \text{for } j = 2, 3, \ldots, p \tag{1.14}$$

and

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \leq 0 \\ 1, & \text{if } \Delta^1 \beta_j > 0 \end{cases}, \quad \text{for } j = p+1, p+2, \ldots, d. \tag{1.15}$$

1.1.6 Valley constraint

Valley behavior can be introduced by the same approach as above by multiplying the data with -1 or by always doing the inverse operation. Therefor, we use the matrix form of the first-order finite difference operator, see 1.6, as mapping matrix $\mathbf{D}$ and define the diagonal elements of the weighting matrix $\mathbf{V}$ as

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \leq 0 \\ 1, & \text{if } \Delta^1 \beta_j > 0 \end{cases}, \quad \text{for } j = 2, 3, \ldots, p \tag{1.16}$$

and

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \geq 0 \\ 1, & \text{if } \Delta^1 \beta_j < 0. \end{cases}, \quad \text{for } j = p+1, p+2, \ldots, \tag{1.17}$$

for p being the index of the B-spline basis function $B_p^l(x)$ with the maximal value at $x^{(\min)}$ with $\min = min_i y^{(i)}$ for$i = 1, 2, \ldots, n$.

1.1.7 Jamming constraint

Jamming the function $f(x)$ by some point $p = \{x^{(jamm)}, y^{(jamm)}\}$ means that the estimated function $f(x^{(jamm)}) \approx y^{(jamm)}$. This can be incorporated using the B-spline basis matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$ as mapping matrix $\mathbf{D}_c$ and a weighting matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ given by

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } x^{(j)} \neq x^{(jamm)} \\ 1, & \text{if } x^{(j)} = x^{(jamm)} \end{cases} \quad \text{for } j = 1, 2, \ldots, n. \tag{1.18}$$

1.1.8 Boundedness constraint

The user-defined constraint for boundedness from below by $M = 0$ uses as mapping matrix $\mathbf{D}$ the B-spline basis matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$. The weighting matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$, with individual weights v_j, is specified as follows:

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } f(x^{(j)}) \geq M \\ 1, & \text{if } f(x^{(j)}) < M \end{cases} \quad \text{for } j = 1, 2, \ldots, n. \tag{1.19}$$

Using different values of M allows us to bound from below from any number M. Switching the comparison operators in (1.19) enables us to bound functions from above.

1.2 Univariate Function Approximation