Chapter 2

Weber Jakob

November 26, 2020

Contents

Chapter 1

Solution Approach

We are now going to 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 work 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

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

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

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

where $\text{pen}(\boldsymbol{\beta})$ specifies a smoothness penalty term. Finally, we are going to incorporate a priori domain knowledge into the fitting process using the approach given by Hofner in [**?**] and apply it to uni- and bivariate functions, i.e. we solve the optimization problem

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

where $\text{con}(\boldsymbol{\beta})$ specifies the user-defined constraint to incorporate a priori domain knowledge with. Various types a priori domain knowledge can be incorporated using the constraints listed in Table 1.2.

Constraint		**Description**	**Section**
Jamming		$f(x^{(M)}) \approx y^{(M)}$	12
Boundedness	lower	$\|f(x)\| \geq M$	12
	upper	$\|f(x)\| \geq M$	12
Monotonicity	increasing	$f'(x) \geq 0$	23
	decreasing	$f'(x) \leq 0$	23
Curvature	convex	$f''(x) \geq 0$	33
	concave	$f''(x) \leq 0$	33
Unimodality	peak	$m = \arg\max_x f(x)$ $f'(x) \geq 0 \quad \text{if } x < m$ $f'(x) \leq 0 \quad \text{if } x > m$	44
	valley	$m = \arg\min_x f(x)$ $f'(x) \leq 0 \quad \text{if } x < m$ $f'(x) \geq 0 \quad \text{if } x > m$	44

Table 1.2: Overview of the considered constraints

1.1 Univariate Function Approximation

To test the methods listed in Table 1.1 and the incorporation of a priori domain knowledge, we use the data

$$D_{univariate} = \{(x^{(i)}, y^{(i)}), i = 1, 2, ...n\}, \tag{1.4}$$

generated from random sampling of $x \in [0, 1]$ of the univariate function $f_1(x)$ given by

$$f_1(x) = 3\sin(3\pi x) + 17x + 3 + \epsilon_i. \tag{1.5}$$

where ϵ_i is Gaussian noise given by $\mathcal{N}(0, 0.1)$. The data is shown in fig:test$_f unc$.

The function is partially increasing. The test function for the two dimensional case is given in Chapter 1.3.4.

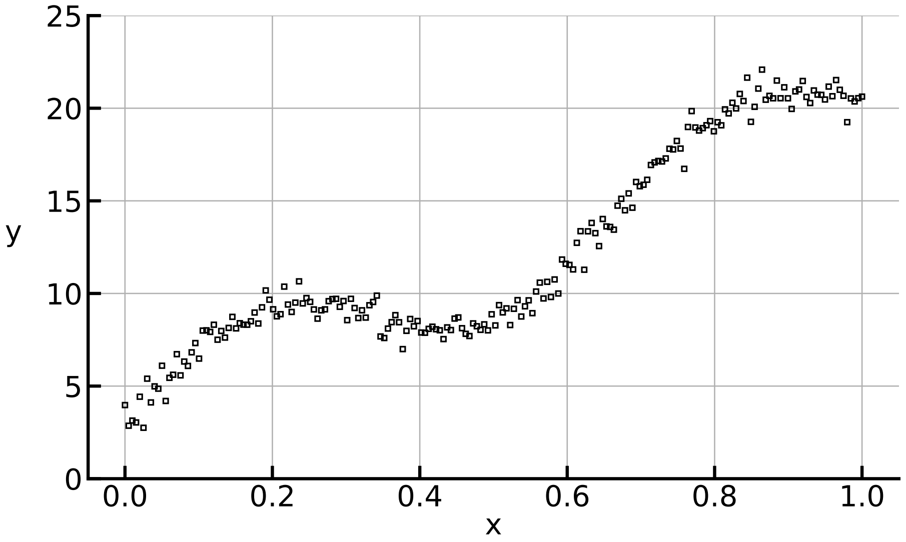

Figure 1.1: Noisy samples determined from test function (1.1)

1.1.1 1d Function Estimation

The goal is to model given data

$$\{\mathbf{x}, \mathbf{y}\} = \{x^{(i)}, y^{(i)}\}, \ i = 1, 2, \ldots, n \tag{1.6}$$

using B-splines as basis functions. Therefore, we want to estimate the unknown function $\mathbf{y} = f(\mathbf{x})$, which can be represented as a linear combination of k B-spline basis functions B_j^m of degree $m = 3$, cf. (**??**), as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}, \tag{1.7}$$

where $\mathbf{X} \in \mathbb{R}^{n \times k}$ is the B-spline basis matrix, cf. (**??**), and $\boldsymbol{\beta} \in \mathbb{R}^k$ are the coefficients to be estimated.

The least squares objective function to be minimized using the complete data is then given by

$$Q_1(\mathbf{y}, \boldsymbol{\beta}) = \|\mathbf{y} - f(\mathbf{x})\|^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{1.8}$$

The coefficients are determined ive function Q_1 given in (1.7) with respect to $\boldsymbol{\beta}$, i.e.

$$\hat{\boldsymbol{\beta}}_{LS} = \arg\min_{\boldsymbol{\beta}} Q_1(\mathbf{y}, \boldsymbol{\beta}). \tag{1.9}$$

Using the least squares algorithm LS, see Chapter **??**, the minimization problem (1.8) yields

$$\hat{\boldsymbol{\beta}}_{LS} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{1.10}$$

Figure 1.2 shows a B-spline model using $k = 10$ splines on an equidistant grid approximating the noisy data presented in Figure 1.1, as well as the individual cubic ($m = 3$) B-spline basis functions $B_i^3(\mathbf{x})$ multiplied with the corresponding, estimated coefficients $\hat{\boldsymbol{\beta}}_{LS,j}$ $j = 1, \ldots, 10$.

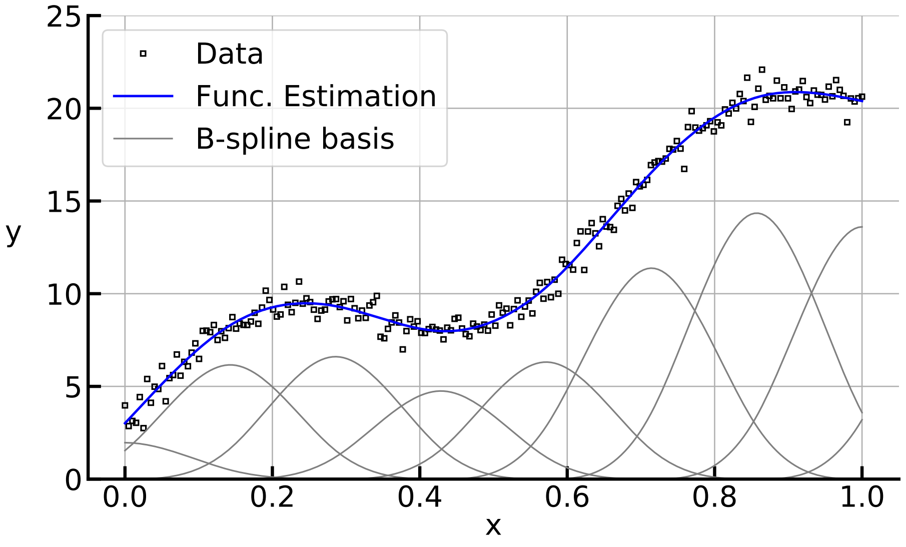

Figure 1.2: Approximation of the noisy data by B-splines without constraints

Note, the number of splines k has a strong influence on the amount of smoothing. A small number k leads to a very smooth estimate, but a large data error. On the other hand, when the number of splines is relatively large, the data error is very small but the smoothness of the estimate is poor. This behavior is an example of the well-known bias-variance dilemma and depicted in Figure 1.3. [?] Here, two B-splines models with $k = 10$ and $k = 50$ are illustrated, which are applied to the noisy data shown in Figure 1.1. To overcome this challenges, the B-splines will be extended by penalizing the second derivative of the estimation, see Chapter 1.1.2.

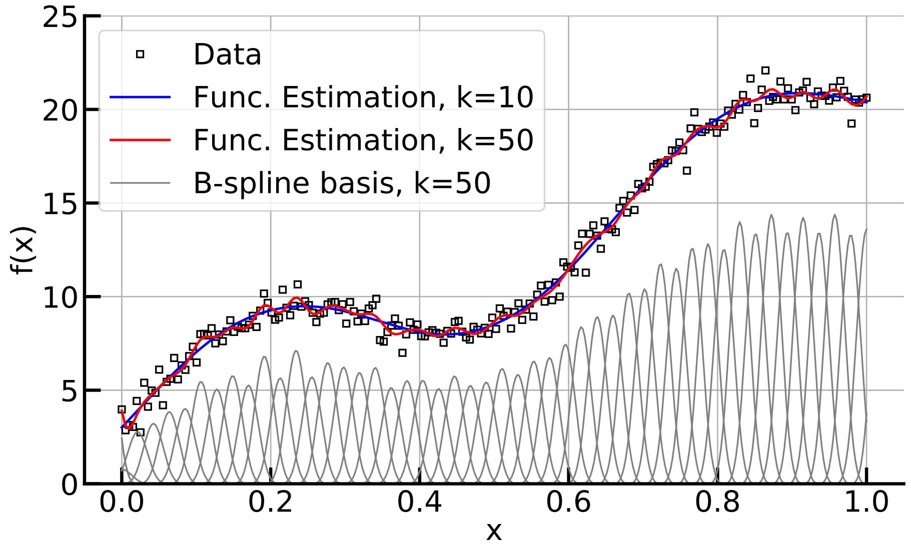

Figure 1.3: Approximation of the noisy data by 10 and 50 B-splines without constraints

1.1.2 1d Smooth Function Estimation

The second derivative of the estimated function $f(x)$, i.e. $f''(x) = \sum_{j=1}^{k} B_i''(x)\beta_k$, has to be penalized to realize a smoother estimate when using a high number of splines. Eilers and Marx have introduced the so-called P-splines. [?], see Chapter **??**. Therefore, the objective function in (1.7) is extended by an additional term considering the smoothness, i.e.

$$Q_2(\mathbf{y}, \boldsymbol{\beta}) = Q_1(\mathbf{y}, \boldsymbol{\beta}) + \lambda_s \mathcal{J}_s(\boldsymbol{\beta}; d) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_s \boldsymbol{\beta}^{\mathrm{T}} \mathbf{D}_d^{\mathrm{T}} \mathbf{D}_d \boldsymbol{\beta}, \tag{1.11}$$

with the smoothing parameter λ_s and an appropriate mapping matrix $\mathbf{D}_d$ capturing the second derivative, which itself is a measure for function wiggliness. Here, an approximation of the second derivative can be performed by the squared finite difference of order d of adjacent coefficients using the matrix form $\mathbf{D}_d$ of the difference operator of order d, see Chapter **??**.

By minimizing the objective function (1.10), i.e.

$$\hat{\boldsymbol{\beta}}_{PLS} = \arg\min_{\boldsymbol{\beta}} Q_2(\mathbf{y}, \boldsymbol{\beta}), \tag{1.12}$$

using the penalized least squares algorithm PLS, the penalized least squares estimates are given by

$$\hat{\boldsymbol{\beta}}_{PLS} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda_s \mathbf{D}_d^{\mathrm{T}} \mathbf{D}_d)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y}. \tag{1.13}$$

In (1.12), the smoothing parameter λ_s plays a critical role and can be optimized using the information criteria specified in Chapter **??**, e.g. AIC and BIC, or by using cross-validation techniques, see Chapter **??**. [?]

For small values $\lambda_s \to 0$, the penalized least squares estimate $\hat{\boldsymbol{\beta}}_{PLS}$ approaches the least squares estimate $\hat{\boldsymbol{\beta}}_{LS}$, cf. (1.9), while for large values $\lambda_s \gg 0$, the fitted function shows the behavior of a polynomial with $d-1$ degrees of freedom. For example, using $d = 2$ and a large smoothing parameter λ_s this configuration leads to a linear function, while using $d = 1$ would lead to a constant function. [**?**]

Figure 1.4 shows the behavior of P-splines of degree $m = 3$ using $k = 50$ splines for several values of the smoothing parameter $\lambda_s = \{10^{-2}, 10^2, 10^5, 10^6\}$ and a smoothness penalty of order $d = 2$. As the value of λ_s gets larger, the fitted curve becomes more smooth and thus the 2^{nd} derivative of the curve becomes smaller due to the penalty considered in the estimation, see (1.12). For very large values of λ_s, the estimate approaches a straight line, see the yellow curve in Figure 1.4.

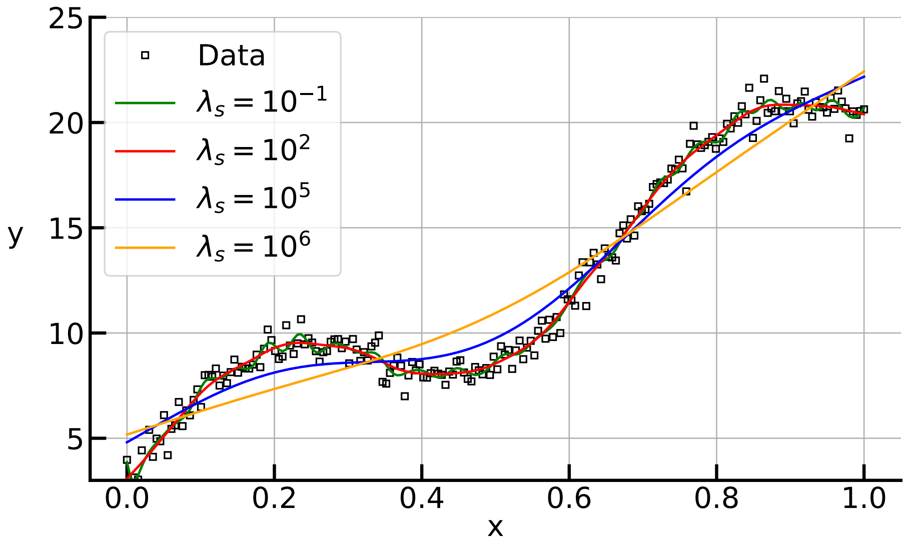

Figure 1.4: Smooth function estimation for different smoothing parameters λ_s

1.1.3 1d Constraint Function Estimation

A priori domain knowledge can now be systematically considered by the extension of the objective function (1.10) using an additional term representing the user-defined constraint, see Table 1.2. Note that this approach incorporates the a priori knowledge as soft constraints. Therefore, no guarantee can be given that the fit holds the constraint for every possible input. The constraint penalized least-squares objective function is given by

$$Q_3(\mathbf{y}, \boldsymbol{\beta}) = Q_1(\mathbf{y}, \boldsymbol{\beta}) + \lambda_s \mathcal{J}_s(\boldsymbol{\beta}; d) + \lambda_c \mathcal{J}_c(\boldsymbol{\beta}; c) \tag{1.14}$$

with the corresponding constraint parameter λ_c, which determines the influence of the user-defined constraint. Note that the parameter λ_c has to be set quite large, i.e. $\lambda_c > 10^4$, compared to λ_s to enforce the user-defined constraint.

Constraints for monotonicity, curvature, unimodality, boundedness and jamming can be modeled as

$$\mathcal{J}_c(\boldsymbol{\beta}; c) = \boldsymbol{\beta}^{\mathrm{T}} \mathbf{D}_c^{\mathrm{T}} \mathbf{V} \mathbf{D}_c \boldsymbol{\beta} \tag{1.15}$$

with the mapping matrix $\mathbf{D}_c$ and the diagonal weighting matrix $\mathbf{V} := \mathbf{V}(\boldsymbol{\beta}; c)$ capturing if the constraint c is active or inactive. The matrices $\mathbf{D}_c$ and $\mathbf{V}$ will further be defined in Chapter 1.2.

By minimizing the objective function (1.13), i.e.

$$\hat{\boldsymbol{\beta}}_{PLS,c} = \arg\min_{\boldsymbol{\beta}} Q_6(\mathbf{y}, \boldsymbol{\beta}), \tag{1.16}$$

the constraint penalized least-squares estimate can be given as

$$\hat{\boldsymbol{\beta}}_{PLS,c} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda_s \mathbf{D}_d^{\mathrm{T}} \mathbf{D}_d + \lambda_c \mathbf{D}_c^{\mathrm{T}} \mathbf{V} \mathbf{D}_c)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y}. \tag{1.17}$$

Note, (1.16) is a nonlinear equation because the matrix $\mathbf{V}$ depends on $\boldsymbol{\beta}$. Thus, it has to be solved iteratively to calculate optimal coefficients $\hat{\boldsymbol{\beta}}_{PLS,c}$. The algorithm is shown in Figure 1.5.

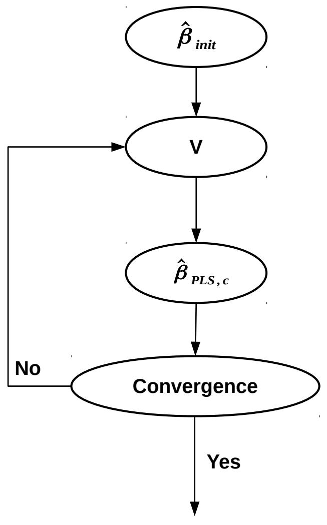
$\hat{\boldsymbol{\beta}}_{init}$
V
$\hat{\boldsymbol{\beta}}_{PLS,c}$
No
Convergence
Yes

Figure 1.5: Penalized iteratively reweighted least squares algorithm

The initial estimate $\hat{\boldsymbol{\beta}}_{init}$ needed to compute the weighting matrix $\mathbf{V}$ is given by the least squares estimate $\hat{\boldsymbol{\beta}}_{LS}$. Based on the initial estimate $\hat{\boldsymbol{\beta}}_{init}$, the weighting matrix $\mathbf{V}$ and then the constraint least-squares estimate $\hat{\boldsymbol{\beta}}_{PLS,c}$ are calculated. The algorithm is performed until no more changes in the weighting matrix $\mathbf{V}$ appear. This scheme is called the penalized iteratively reweighted least squares and is abbreviated by PIRLS. [**?**]

Figure 1.6 shows an example, where the noisy data shown in Figure 1.1 is approximated by considering the monotonicity constraint. The estimate has to be monotonically increasing in contrast to the data, i.e. $f'(x) \geq 0$. The smoothing parameter λ_s was optimized using cross-validation and set to $\lambda_s = 271.9$. The constraint parameter λ_c was set to $\lambda_c = 6000$. For both function estimations, i.e. using P-splines (blue curve) vs. using constraint P-splines (red curve), the number of used splines k was set to $k = 30$.

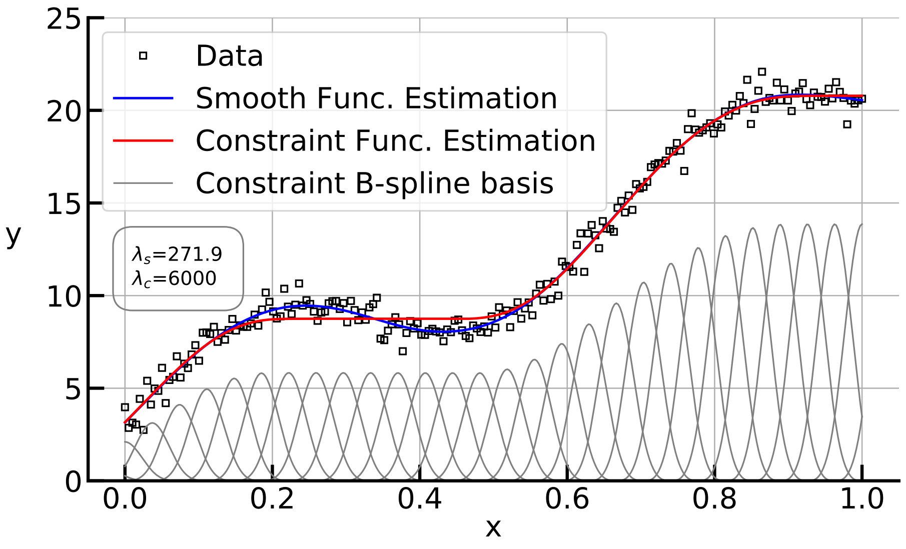

Figure 1.6: Approximation of the noisy data by P-splines and P-splines with the monotonic increasing constraint

The constraint function estimation (red curve in Figure 1.6), follows the monotonicity constraint far better that the smooth function estimation (blue curve in Figure 1.6). For $x < 0.15$ and $x > 0.6$, the two fits are nearly identical, since no constraint violation is present. Note, the entries of the weighting matrix $\mathbf{V}$ in this region are therefore 0 because the constraint is not active. For $x \in [0.15, 0.6]$ the constraint is active. The red fit produces an almost constant line in this region as an optimal solution for the competing goals of data accuracy, smoothness and constraint fidelity.

This shows, that the incorporation of a priori knowledge in the fitting process using P-splines is in principle possible using an appropriate choice of the mapping matrix $\mathbf{D}_c$ and the weighting matrix $\mathbf{V}$ as well as an iterative fitting approach using penalized iteratively reweighted least squares. These matrices are futher discussed in the following chapters.

1.2 User-defined Constraints

As stated before, a priori domain knowledge given in Table 1.2 can be introduced by the choice of the mapping matrix $\mathbf{D}_c$ and the weighting matrix $\mathbf{V}$, cf. (1.14) and (1.16). Now a description of the different matrices, which are used to enforce the a priori known domain behavior, is presented.

1.2.1 Monotonicity Constraint

The mapping matrix $\mathbf{D}_{monoton}$ enforcing monotonic behavior is given by the first order difference operator Δ^1 for equidistant knot placement, cf. **??**. The corresponding matrix for k splines is given as

$$\mathbf{D}_{monoton} = \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \end{pmatrix} \in \mathbb{R}^{k-1 \times k}. \tag{1.18}$$

The difference between monotonic increasing and decreasing behavior is controlled by the weighting matrix $\mathbf{V}$. For increasing behavior, the weighting matrix $\mathbf{V}$ is given by the weights v_j according to

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \geq 0 \\ 1, & \text{if } \Delta^1 \beta_j < 0 \end{cases} \quad \text{for } j = 2, \dots, k-1. \tag{1.19}$$

For decreasing behavior, the weighting matrix $\mathbf{V}$ is given by the weights v_j according to

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1 \beta_j \leq 0 \\ 1, & \text{if } \Delta^1 \beta_j > 0 \end{cases} \quad \text{for } j = 2, \dots, k-1. \tag{1.20}$$

This states, that the penalty term $\mathcal{J}_c(\boldsymbol{\beta}; c)$ only contributes if adjacent coefficients β_{j-1} and β_j are increasing or decreasing, respectively. [**?**] [**?**]

1.2.2 Curvature Constraint

In the simplest case, the curvature of the function $f(x)$ can either be convex, i.e. $f''(x) \geq 0$, or concave, i.e. $f''(x) \leq 0$. The mapping matrix $\mathbf{D}_{curvature}$ enforcing this behavior can be approximated by the second order difference operator Δ^2 for equidistant knot placement, cf. (**??**). The corresponding matrix for k splines is given as

$$\mathbf{D}_{curvature} = \begin{pmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \in \mathbb{R}^{k-2 \times k}. \tag{1.21}$$

The difference between concave and convex curvature is controlled by the weighting matrix $\mathbf{V}$. For concave curvature, the weighting matrix $\mathbf{V}$ is given by the weights v_j according to

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^2\beta_j \leq 0 \\ 1, & \text{if } \Delta^2\beta_j > 0 \end{cases} \quad \text{for } j = 1, \ldots, k-2. \tag{1.22}$$

For convex curvature, the weighting matrix $\mathbf{V}$ is given by the weights v_j according to

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^2\beta_j \geq 0 \\ 1, & \text{if } \Delta^2\beta_j < 0 \end{cases} \quad \text{for } j = 1, \ldots, k-2. \tag{1.23}$$

Therefore, the penalty term $\mathcal{J}_c(\boldsymbol{\beta}; c)$ in (1.13) or (1.16) only contributes if the second order difference of adjacent coefficients $\boldsymbol{\beta}$ is either positive or negative, respectively. [**?**]

1.2.3 Unimodality Constraint

We assume that there is a peak in the data $\{x^{(i)}, y^{(i)}\}$ and therefore want to constrain the fit to include a peak. The peak constraint is given by the unimodal mapping matrix $D_{unimodal}$ and the peak weighting matrix V. A function $f(x)$ is said to be unimodal if for some value m, it is monotonically increasing for $x \leq m$ and monotonically decreasing for $x \geq m$.

The mapping matrix $\mathbf{D}_{unimodal}$ enforcing unimodal behavior can be constructed using the first order difference operator Δ^1 for equidistant knot placement, cf. (**??**), and is given for k splines as

$$\mathbf{D}_{unimodal} = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{k-1 \times k} \tag{1.24}$$

The weighting matrix $\mathbf{V}$ now has a special structure. First, we construct the B-spline basis using the given data as in Chapter **??**. We then need to find the index j_{peak} of the *peak spline*, which has the maximal value at the peak data point $\max\{f(x^{(i)}) \ \forall \ i\}$, see Figure 1.7. The index j_{peak} is now used as splitting point for the weighting matrix $\mathbf{V}$. All coefficients β_j for $j < j_{peak}$ are constrained to be monotonic increasing, i.e. $\Delta^1\beta_j \geq 0$ for $j = 1, \ldots, j_{peak} - 1$, while all coefficients β_j for $j > j_{peak}$ are constrained to be monotonic decreasing, i.e. $\Delta^1\beta_j \leq 0$ for $j = j_{peak}+1, \ldots, k$. The coefficient $\beta_{j_{peak}}$ stays unconstrained. [**?**]

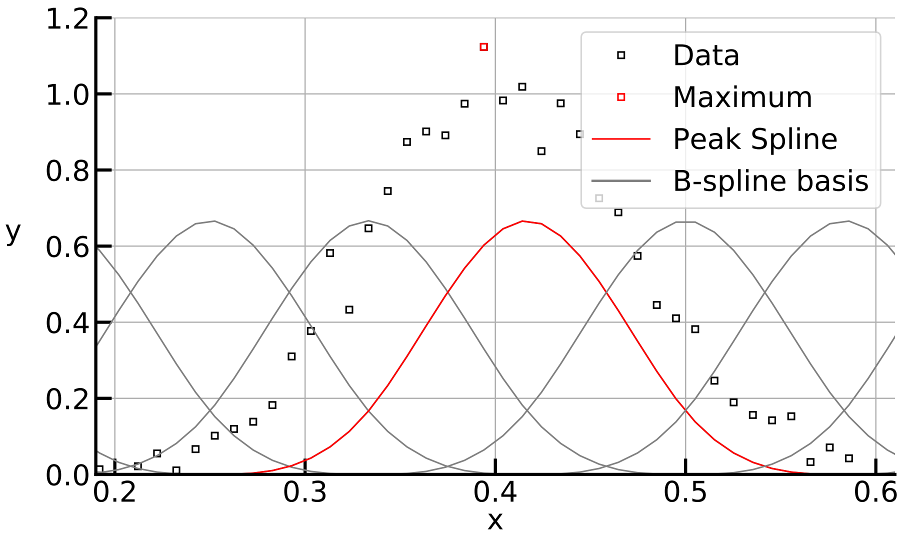

Figure 1.7: Identification of the peak spline based on data

The weights v_j to incorporate the peak constraint have the following structure, i.e.

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1\beta_j \geq 0 \\ 1, & \text{if } \Delta^1\beta_j < 0 \end{cases} \quad , \quad \text{for } j = 2, \ldots, j_{peak} - 1 \tag{1.25}$$

and

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1\beta_j \leq 0 \\ 1, & \text{if } \Delta^1\beta_j > 0 \end{cases} \quad , \quad \text{for } j = j_{peak} + 1, \ldots, k. \tag{1.26}$$

The weight $v_{j_{peak}}$ for the *peak spline* is given by $v_{j_{peak}}(\boldsymbol{\beta}) = 0$.

When assuming a valley in the data, the same approach as above can easily be used by multiplying the data with -1 or by always doing the inverse operation, i.e. finding the index j_{valley} of the *valley spline*, then constraining all splines for $j < j_{valley}$ to be monotonic decreasing, i.e. $\Delta^1\beta_j \leq 0$ for $j = 1, \ldots, j_{valley} - 1$, and all splines for $j > j_{valley}$ to be monotonic increasing, i.e. $\Delta^1\beta_j \geq 0$ for $j = j_{valley} + 1, \ldots, k$. The coefficient $\beta_{j_{valley}}$ stays unconstrained.

The weights v_j to consider a valley constraint are given by

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1\beta_j \leq 0 \\ 1, & \text{if } \Delta^1\beta_j > 0 \end{cases} \quad , \quad \text{for } j = 2, \ldots, j_{valley} - 1 \tag{1.27}$$

and

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } \Delta^1\beta_j \geq 0 \\ 1, & \text{if } \Delta^1\beta_j < 0. \end{cases} \quad , \quad \text{for } j = j_{valley} + 1, \ldots, k. \tag{1.28}$$

The weight $v_{j_{valley}}$ for the *valley spline* is given by $v_{j_{valley}}(\boldsymbol{\beta}) = 0$.

1.2.4 Boundedness Constraint

For certain physical systems, it is known a priori that the measured quantity cannot be smaller than zero, i.e. $f(x) \geq 0$. Using data-driven modeling on noisy data can lead to predictions in the interpolation and extrapolation regime, which may not hold this constraint due to uncertainties captured by the data. It is therefore appropriate to apply the user-defined constraint of boundedness from below.

The user-defined constraint for boundedness from below by $M = 0$ uses as mapping matrix $\mathbf{D}_c$ 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:

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } f(x^{(j)}) \geq M \\ 1, & \text{if } f(x^{(j)}) < M \end{cases} \quad \text{for } j = 1, \ldots, n. \tag{1.29}$$

Using different values of M allows us to bound from below from any number M. Switching the comparison operators in (1.28) enables us to bound functions from above.

1.2.5 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

$$v_j(\boldsymbol{\beta}) = \begin{cases} 0, & \text{if } x^{(j)} \neq x^{(jamm)} \\ 1, & \text{if } x^{(j)} = x^{(jamm)} \end{cases} \quad \text{for } j = 1, \ldots, n. \tag{1.30}$$

1.2.6 Penalty Term for Tensor-Product Splines

To extend the framework of mapping matrices to two dimensions and tensor-product splines, we again use the concept of Kronecker products given in Chapter **??**. In principle, every possible pair of one dimensional user-defined constraints can be constructed using the approach in Chapter **??**, e.g. unimodality in two dimensions would be obtained using the unimodal mapping matrix depicted above for each dimension. We then also need to include the constraint specific weight matrices $\mathbf{V}$.

The penalty term for the constraint given by c_1 for dimension 1 and c_2 for dimension 2 then has the form

$$\mathcal{J}_c(\boldsymbol{\beta}; c) = \boldsymbol{\beta}^{\mathrm{T}} \Big[\mathbf{I}^2 \otimes \mathbf{K}_{c_1} + \mathbf{K}_{c_2} \otimes \mathbf{I}^1 \Big] \boldsymbol{\beta} \tag{1.31}$$

with the respective penalty matrices $\mathbf{K}_{c_1} = \mathbf{D}_{c_1}^{\mathrm{T}} \mathbf{V}_1 \mathbf{D}_{c_1}$ for dimension x_1 and $\mathbf{K}_{c_2} = \mathbf{D}_{c_2}^{\mathrm{T}} \mathbf{V}_2 \mathbf{D}_{c_2}$ for dimension x_2 using the weighting matrices $\mathbf{V}_1$ and $\mathbf{V}_2$, the mapping matrices $\mathbf{D}_{c_1}$ and $\mathbf{D}_{c_2}$ and the identity matrices $\mathbf{I}^1$ and $\mathbf{I}^2$ for the respective dimension.

1.3 n-d Constraint Function Estimation

The extension from one input to multiple input dimensions uses the concept of additive models given in Chapter **??**. Given input data $\{x_1^{(i)}, \dots, x_q^{(i)}, y^{(i)}\},\ i = 1, 2, \dots, n$ and q as the number of inputs, the combined model using all available B-splines and tensor-product splines is given, cf. (**??**), as

$$y = f(x_1, ..., x_q) = \sum_{j=1}^{q} s_j(x_j) + \sum_{j=1}^{q-1} \sum_{r>j}^{q} t_{j,r}(x_j, x_r) \tag{1.32}$$

where $s_j(x_j)$ is the B-spline estimate given by $s_j(x_j) = \mathbf{X}_{s_j}\boldsymbol{\beta}_{s_j}$ and $t_{j,r}(x_j, x_r)$ is the tensor-product estimate is given by $t_{j,r}(x_j, x_r) = \mathbf{X}_{t_{j,r}}\boldsymbol{\beta}_{t_{j,r}}$. The number of individual estimates is given by

$$n_{total} = q + \frac{q(q-1)}{2}. \tag{1.33}$$

The constrained penalized least squares objective function for additive models can now be written similar to (1.13) as

$$Q_6(\mathbf{y}, \boldsymbol{\beta}) = Q_1(\mathbf{y}, \boldsymbol{\beta}) + \boldsymbol{\lambda}_s^{\mathrm{T}} \mathcal{J}_s(\boldsymbol{\beta}; \mathbf{d}) + \boldsymbol{\lambda}_c^{\mathrm{T}} \mathcal{J}_c(\boldsymbol{\beta}; \mathbf{c}). \tag{1.34}$$

with $\boldsymbol{\lambda}_s \in \mathbb{R}^{n_{total}}$ and $\boldsymbol{\lambda}_c \in \mathbb{R}^{n_{total}}$ defined as vectors with one value of smoothness and constraint parameter for each individual estimate, respectively.

We now need to specify the three parts of the objective function in (1.33).

1.3.1 Data Term

Assuming the use of k splines for the B-spline estimates and k^2 splines for the tensor-product estimates, the total number of coefficients to be determined is given by

$$k_{total} = qk + \frac{q(q-1)}{2}k^2. \tag{1.35}$$

Since all B-spline and tensor-product spline models follow a linear model structure, see Chapter **??** and **??**, we can combine them into one large model, cf. (**??**), given by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} \tag{1.36}$$

where the matrix $\mathbf{X} \in \mathbb{R}^{n \times k_{total}}$ is given in (**??**) as horizontal concatenation of the individual bases and the combined coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^{k_{total}}$ is given in (**??**) by a vertical concatenation of the individual coefficient vectors.

The data term $Q_1(\mathbf{y}, \boldsymbol{\beta})$ in the constrained penalized least squares objective function given in (1.33) can now be evaluated using arbitrary input dimensions.

1.3.2 Smoothness Term

The combined smoothness penalty term $\mathcal{J}_s(\boldsymbol{\beta}; \mathbf{d}) \in \mathbb{R}^{n_{total}}$ is then given as

$$\mathcal{J}_s(\boldsymbol{\beta}; \mathbf{d}) = \begin{pmatrix} \mathcal{J}_{s_1}(\boldsymbol{\beta}_{s_1}; d_{s_1}) \\ \vdots \\ \mathcal{J}_{s_q}(\boldsymbol{\beta}_{s_q}; d_{s_q}) \\ \mathcal{J}_{t_{1,2}}(\boldsymbol{\beta}_{t_{1,2}}; d_{t_{1,2}}) \\ \vdots \\ \mathcal{J}_{t_{q-1,q}}(\boldsymbol{\beta}_{t_{q-1,q}}; d_{t_{q-1,q}}) \end{pmatrix} \tag{1.37}$$

with $\mathcal{J}_e(\boldsymbol{\beta}_e; d_e) = \boldsymbol{\beta}_e^{\mathrm{T}} \mathbf{D}_{d_e}^{\mathrm{T}} \mathbf{D}_{d_e} \boldsymbol{\beta}_e$ determining the smoothness penalty term using the coefficients $\boldsymbol{\beta}_e$ and mapping matrix $\mathbf{D}_{d_e}$, see Chapter **??** and Chapter **??**, for each estimate $e \in \{s_1, \dots, s_q, t_{1,2}, \dots, t_{q-1,q}\}$. The vector $\mathbf{d} \in \mathbb{R}^{n_{total}}$ consists of the orders d_e determining the mapping matrix $\mathbf{D}_{d_e}$ of the smoothness constraint for each individual estimate e.

1.3.3 Constraint Term

The combined constraint penalty term $\mathcal{J}_c(\boldsymbol{\beta}; \mathbf{c}) \in \mathbb{R}^{n_{total}}$ is then given as

$$\mathcal{J}_c(\boldsymbol{\beta}; \mathbf{c}) = \begin{pmatrix} \mathcal{J}_{s_1}(\boldsymbol{\beta}_{s_1}; c_{s_1}) \\ \vdots \\ \mathcal{J}_{s_q}(\boldsymbol{\beta}_{s_q}; c_{s_q}) \\ \mathcal{J}_{t_{1,2}}(\boldsymbol{\beta}_{t_{1,2}}; c_{t_{1,2}}) \\ \vdots \\ \mathcal{J}_{t_{q-1,q}}(\boldsymbol{\beta}_{t_{q-1,q}}; c_{t_{q-1,q}}) \end{pmatrix} \tag{1.38}$$

with $\mathcal{J}_e(\boldsymbol{\beta}_e; c_e) = \boldsymbol{\beta}_e^{\mathrm{T}} \mathbf{D}_{c_e}^{\mathrm{T}} \mathbf{V}_{c_e} \mathbf{D}_{c_e} \boldsymbol{\beta}_e$ determining the constraint penalty term using the coefficients $\boldsymbol{\beta}_e$, the mapping matrix $\mathbf{D}_{c_e}$ and the weighting matrix $\mathbf{V}_e$ for each estimate $e \in \{s_1, \dots, s_q, t_{1,2}, \dots, t_{q-1,q}\}$, see Chapter (1.2). The vector $\mathbf{c} \in \mathbb{R}^{n_{total}}$ consists of the constraint type c_e, e.g. monoton increasing, determining the mapping matrix $\mathbf{D}_{c_e}$ for each individual estimate e.

The objective function (1.33) is then optimized, i.e.

$$\hat{\boldsymbol{\beta}}_{PLS,c,nd} = \arg\min_{\boldsymbol{\beta}} Q_6(\mathbf{y}, \boldsymbol{\beta}), \tag{1.39}$$

using the penalized iteratively reweighted least squares algorithm, cf. (1.16), to obtain the coefficients $\hat{\boldsymbol{\beta}}_{PLS,c,nd}$ as

$$\hat{\boldsymbol{\beta}}_{PLS,c,nd} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \mathbf{K}_s + \mathbf{K}_c)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{1.40}$$

In (1.39), $\mathbf{X} \in \mathbb{R}^{n \times k_{total}}$ is the combined basis matrix, cf. (**??**), $\mathbf{K}_s \in \mathbb{R}^{k_{total} \times k_{total}}$ is the combined smoothness matrix given as

$$\mathbf{K}_s = \begin{pmatrix} \lambda_{s_1}\mathbf{D}_{d_{s_1}}^{\mathrm{T}}\mathbf{D}_{d_{s_1}} & 0 & & & & \\ 0 & \ddots & 0 & & & \\ & 0 & \lambda_{s_q}\mathbf{D}_{d_{s_q}}^{\mathrm{T}}\mathbf{D}_{d_{s_q}} & 0 & & \\ & & 0 & \lambda_{s_{1,2}}\mathbf{D}_{d_{t_{1,2}}}^{\mathrm{T}}\mathbf{D}_{d_{t_{1,2}}} & 0 & \\ & & & 0 & \ddots & 0 \\ & & & & 0 & \lambda_{s_{q-1,q}}\mathbf{D}_{d_{t_{q-1,q}}}^{\mathrm{T}}\mathbf{D}_{d_{t_{q-1,q}}} \end{pmatrix} \tag{1.41}$$

and $\mathbf{K}_c \in \mathbb{R}^{k_{total} \times k_{total}}$ is the combined constraint matrix as

$$\mathbf{K}_c = \begin{pmatrix} \lambda_{c_1}\mathbf{D}^{\mathrm{T}}_{cs_1}\mathbf{V}_{cs_1}\mathbf{D}_{cs_1} & 0 & 0 & & & \\ 0 & \ddots & 0 & & & \\ 0 & 0 & \lambda_{c_q}\mathbf{D}^{\mathrm{T}}_{cs_q}\mathbf{V}_{cs_q}\mathbf{D}_{cs_q} & 0 & & \\ & & 0 & \lambda_{c_{1,2}}\mathbf{D}^{\mathrm{T}}_{ct_{1,2}}\mathbf{V}_{ct_{1,2}}\mathbf{D}_{ct_{1,2}} & 0 & 0 \\ & & & 0 & \ddots & 0 \\ & & & 0 & 0 & \lambda_{c_{q-1,q}}\mathbf{D}^{\mathrm{T}}_{ct_{q-1,q}}\mathbf{V}_{ct_{q-1,q}}\mathbf{D}_{ct_{q-1,q}} \end{pmatrix}. \tag{1.42}$$

1.3.4 2-d Example

As example for the n-d constraint function estimation, we take a look at the function

$$f(x_1, x_2) = 2\exp\Big(-\frac{(x_1 - 0.25)^2}{0.08}\Big) + x_2^2 + \eta \tag{1.43}$$

for $x_1 \in [0,1]$ and $x_2 \in [0,1]$ and random Gaussian noise η with $\sigma_{noise} = 0.1$. Therefore we expect a peak in dimension x_1 as well as increasing behavior for dimension x_2, see Figure 1.8. The user-defined constraints are therefore c_1 = unimodal and c_2 = monotonic increasing Using this knowledge, we create a model with the following characteristics:

- B-spline smooth $s_1(x_1)$: $k_{x_1} = 50$, $c = c_1$, $\lambda_s = 1$ and $\lambda_c = 6000$
- B-spline smooth $s_2(x_2)$: $k_{x_2} = 50$, $c = c_2$, $\lambda_s = 1$ and $\lambda_c = 6000$

The fit for this model as well as the individual estimates $s_1(x_1)$ and $s_2(x_2)$ are shown in Figure 1.8. The model fits the data quite well and holds the specified constraints for the individual dimensions.

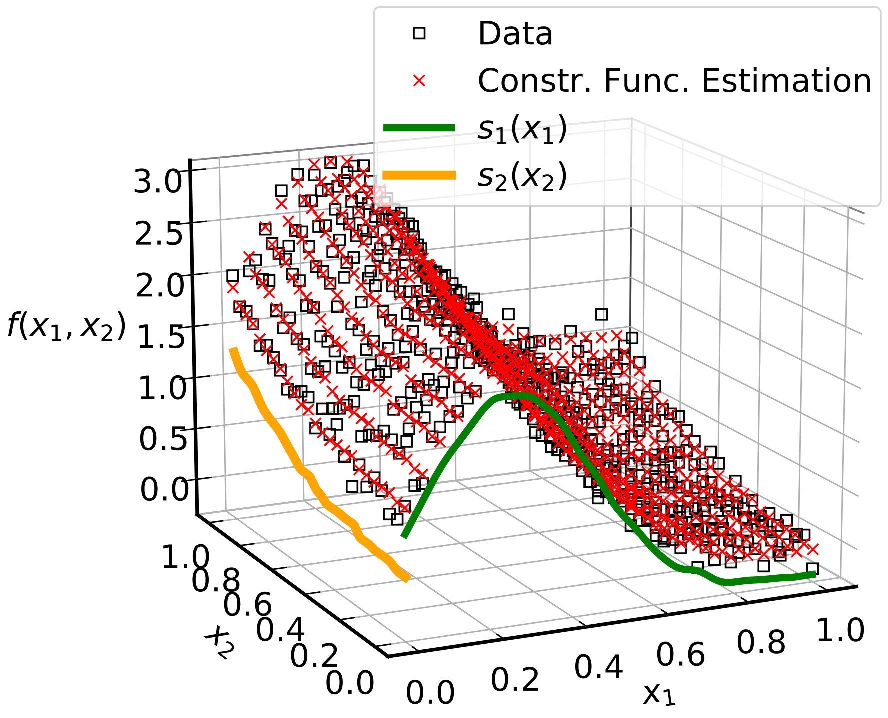

Figure 1.8: 2-d test function for n-d constrained function estimation