Knowl Inf Syst (2019) 59:197–218
https://doi.org/10.1007/s10115-018-1201-2

REGULAR PAPER

A smoothed monotonic regression via L2 regularization

Oleg Sysoev[1] · **Oleg Burdakov**[2]

Received: 8 June 2017 / Revised: 21 February 2018 / Accepted: 18 April 2018 /
Published online: 26 April 2018

Abstract Monotonic regression is a standard method for extracting a monotone function from non-monotonic data, and it is used in many applications. However, a known drawback of this method is that its fitted response is a piecewise constant function, while practical response functions are often required to be continuous. The method proposed in this paper achieves monotonicity and smoothness of the regression by introducing an L2 regularization term. In order to achieve a low computational complexity and at the same time to provide a high predictive power of the method, we introduce a probabilistically motivated approach for selecting the regularization parameters. In addition, we present a technique for correcting inconsistencies on the boundary. We show that the complexity of the proposed method is $O(n^2)$. Our simulations demonstrate that when the data are large and the expected response is a complicated function (which is typical in machine learning applications) or when there is a change point in the response, the proposed method has a higher predictive power than many of the existing methods.

Keywords Monotonic regression · Kernel smoothing · Penalized regression · Probabilistic learning

1 Introduction

There is a multitude of applications in which it can be believed that one or more predictors have a monotonic relationship with the response variable, i.e., the response function is non-decreasing (or non-increasing) when one or more of these predictors increase. Such applications can be found in psychology, biology, signal processing, economics and many

✉ Oleg Sysoev
oleg.sysoev@liu.se

[1] Department of Computer and Information Science, Linköping University, 58183 Linköping, Sweden

[2] Department of Mathematics, Linköping University, 58183 Linköping, Sweden

other disciplines [1,2,21,22]. Perhaps the most widely known approach which allows for extracting a monotonic dependence is monotonic regression (MR) [5,38].

In this paper, we consider the case of a model with one predictor $x \in R$. The corresponding MR problem can be formulated as follows. Let $f(x)$ be an unknown monotonic true response function. Given n observations $(X_1, Y_1), \ldots, (X_n, Y_n)$, where $X_i \in R$ and $Y_i \in R$ are the observed predictor and response values, respectively. The values of Y_i are random with mean $f(X_i)$. The predictor values are sorted as $X_1 < X_2 < \ldots < X_n$. It is required that the fitted response values inherit this monotonicity. They are obtained by solving in $\mu \in R^n$ the optimization problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} (\mu_i - Y_i)^2 \\ \text{s.t. } & \mu_i \leq \mu_{i+1}, \; i = 1, \ldots, n-1, \end{aligned} \tag{1}$$

It can be easily shown that a solution to this problem is a maximum likelihood estimator of the parameters μ_i, when Y_i are assumed to be normally distributed with mean μ_i and, at the same time, the μ_is are non-decreasing with respect to the predictor.

There exists a simple algorithm, called pool adjacent violators (PAV), that solves problem (1) in $O(n)$ steps [4]. It is widely used due to its computational efficiency, but yet some practitioners are skeptic about using MR in their applications because the fitted response returned by the MR resembles a step function while the expected response in these applications is believed to be continuous and smooth. When there is more than one predictor involved, MR can also be applied, and there are computationally heavy exact algorithms for solving MR problem, such as the algorithm in Maxwell and Muckstadt [25], or computationally inexpensive approximate MR algorithms, such as algorithms in Burdakov et al. [11] and Sysoev et al. [40]. However, the fitted response still resembles a piecewise constant function even in the model with many predictors. Recently, a new generalized additive framework was proposed in Chen and Samworth [12], but when the model has one input, this method returns the same response as the PAV algorithm.

In order to handle the lack of smoothness of the MR, various techniques were developed for the model with one predictor. One group of methods [17,19,24,28] combines the MR approach with kernel smoothers, and the fitted response appears to be monotonic and smooth. A particularly simple ad hoc approach is discussed in Mammen [24], where a smooth monotonic estimator m_{IS} first applies the monotonic (called isotonic in Mammen [24]) regression and then a kernel smoother; in another estimator m_{SI}, the smoothing is followed by the monotonization. Since the complexity of the kernel smoothing is $O(n^2)$ [16] and the complexity of the MR computation is $O(n)$, the computational cost of m_{IS} and m_{SI} is $O(n^2)$. Other methods in this group seem to be more computationally heavy: They either are based on a numerical integration or apply quadratic programming.

Another large group of methods produces a smooth monotonic fit by applying smoothing spline techniques [8,9,26,27,35,36,44]. The quality of the fitting and prediction, as well as computational time of these methods, depends on the number of knots in a spline. If the number of knots (which defines both the complexity and the quality of the model) is fixed, these models are capable to fit quite large data within an acceptable time. However, modern applications often involve data that are large and at the same time are generated by complicated processes. In order to fit such large and complex data, a large number of knots might be needed, which in its turn leads to prohibitively large computational times. We discuss this issue further in Sect. 5. In addition, we include into our simulation studies the method introduced in Pya and Wood [35] (which was shown to be more precise than any of Bollaerts et al. [8], Meyer [26], Meyer [27]).

In the literature, there is also some amount of work based on Bayesian approaches (see for ex. [20,23,30,31,37,39,42]). In general, these methods involve Markov chain Monte Carlo sampling or other type of stochastic optimization, which makes them computationally heavy compared to the aforementioned frequentist alternatives. Complexity and scalability issues are often dismissed or not in the main focus in these papers, and the authors often consider pretty small samples in their simulations. For instance, the method described in Holmes and Heard [20] was presented as ‘extremely fast to simulate,’ and it completes the computations ‘under two minutes’ for a sample size containing around 200 observations. In contrast, the method presented in this paper can fit more than a million observations in less than a minute, as it is illustrated in Burdakov and Sysoev [10]. However, in order to compare the predictive power of these models and our approach, we include in our simulation studies the approach described in Neelon and Dunson [30,31]. The choice of the method was motivated by the public availability of the code [32], and also by the results of the simulation in Shively et al. [39] demonstrating that there is no obvious winner among the methods in Holmes and Heard [20], Neelon and Dunson [31] and Shively et al. [39]. We have also decided not to include the methods based on Gaussian process optimization (such as [23,37,42]) into our simulations because they require at least $O(n^3)$ numerical operations per iteration, which makes these algorithms too expensive for large data.

The purpose of this work is to develop a method which (a) is fast and well scalable, i.e., able to efficiently handle large data sets and can be efficiently applied in iterative algorithms like GAM or gradient boosting; (b) is statistically motivated rather than ad hoc; (c) enables a user to select a desired degree of smoothness, as it is with splines or kernel smoothers and (d) has a reasonably good predictive power.

For achieving these goals, we modify the MR formulation (1) by penalizing the difference between adjacent fitted response values. This is aimed at eliminating sharp ‘jumps’ of the response function. In this paper, we show that such a simple idea has a rigorous probabilistic background. Our smoothed monotonic regression (SMR) problem is formulated as

$$\begin{aligned} \min \; & \sum_{i=1}^{n} (\mu_i - Y_i)^2 + \sum_{i=1}^{n-1} \lambda_i \, (\mu_{i+1} - \mu_i)^2 \\ \text{s.t.} \quad & \mu_i \le \mu_{i+1}, \; i = 1, \ldots, n-1, \end{aligned} \tag{2}$$

where $\lambda_i > 0, i = 1, \ldots, n-1$, are preselected penalty parameters whose choice is discussed in this paper.

In this work and in an accompanying paper [10], we study problem (2) from different perspectives and present a smoothed pool adjacent violators (SPAV) algorithm that computes the solution of the SMR problem. Paper published by Burdakov and Sysoev [10] is focused on convergence properties given a set of penalty parameters $\Lambda = \{\lambda_1, \ldots, \lambda_{n-1}\}$ and contains the proof of the optimality of the SPAV algorithm from the optimizational perspective. However, the algorithm presented in Burdakov and Sysoev [10] cannot be directly used in machine learning applications. Firstly, the algorithm requires the set Λ to be specified, and if traditional parameter selection, i.e., cross-validation, is used, the overall computational time becomes exponential in n (because at least 2^{n-1} possible Λ need to be considered). Secondly, the algorithm is only able to deliver predictions for the values of the input variables present in the training data; the prediction for all other input values becomes undefined. This is a known limitation of the monotonic regression discussed in Sysoev et al. [41].

The current paper introduces a probabilistic framework that allows for resolving the critical issues mentioned above and thus enables employing the SMR in machine learning applications. In particular, this framework induces a motivated choice for the penalty parameters that reduces the overall worst-case computational cost to $O(n^2)$. This framework makes it also possible to deliver consistent predictions for the input values that are outside the training data. In addition, we present a technique that performs correction of the boundary inconsistencies. Last but not least, the paper presents numerical analysis of the predictive performance of the proposed model and some alternative algorithms.

The paper is organized as follows. Section 2 presents the SPAV algorithm and also contains estimates of the complexity of this algorithm given a fixed set of Λ. In Sect. 3, a probabilistic model inducing the SMR problem is presented, an appropriate choice of the penalty parameters is discussed and a consistent predictive model is presented. In particular, this section introduces a generalized cross-validation for the probabilistic model corresponding to (2). Section 4 considers a problem of inconsistency of the SMR on the boundaries and introduces a correction strategy. In Sect. 5, results of numerical simulations and a comparative study of our smoothing approaches and some alternative algorithms are presented. Section 6 contains conclusions.

2 SPAV algorithm

One can see that the objective function in (2) is strictly convex, because the first term is a strictly convex function and the other one is convex. The constraints are linear. Therefore, the SMR problem (2) is a strictly convex quadratic programming (QP) problem. In principle, it can be solved by the conventional QP algorithms [33]. However, these algorithms do not make use of the special structure of the objective function or constraints in (2), whereas taking this structure into account, as one can see below, allows for substantial speeding-ups. It should also be noted that the general purpose QP algorithms are not of polynomial complexity in contrast to our algorithm.

Owing to the strict convexity of the SMR problem, its solution is unique and it is determined by the Karush–Kuhn–Tucker (KKT) optimality conditions [33]. To derive them, consider the Lagrangian function

$$L(\mu, \rho) = \sum_{i=1}^{n} (\mu_i - Y_i)^2 + \sum_{i=1}^{n-1} \lambda_i (\mu_{i+1} - \mu_i)^2 - \sum_{i=1}^{n-1} \rho_i (\mu_{i+1} - \mu_i),$$

where $\rho \in R^{n-1}$ is the vector of nonnegative Lagrange multipliers. Its first derivative is calculated by the formula

$$\nabla_{\mu_i} L(\mu, \rho) = \begin{cases} 2(\mu_1 - Y_1) + 2\lambda_1 (\mu_1 - \mu_2) + \rho_1, \ i = 1, \\ 2(\mu_i - Y_i) + 2\lambda_{i-1} (\mu_i - \mu_{i-1}) + \\ \quad +2\lambda_i (\mu_i - \mu_{i+1}) - (\rho_{i-1} - \rho_i), \ i = 2, \ldots, n-1, \\ 2(\mu_n - Y_n) + \\ \quad + 2\lambda_{n-1} (\mu_n - \mu_{n-1}) - \rho_{n-1}, \qquad i = n. \end{cases} \tag{3}$$

To represent the KKT conditions in a compact form, we shall use the following notation. Let

$$I(a) = \begin{cases} 1, \ \text{if } a \text{ is true}, \\ 0, \ \text{if } a \text{ is false} \end{cases}$$

stand for the indicator function. We denote by $A \in R^{n\times n}$ and $B \in R^{n\times(n-1)}$ the matrices whose elements are defined as

$$A_{ij} = \begin{cases} 1 + \lambda_{i-1} I(i > 1) + \lambda_i I(i < n), & \text{if } j = i, \\ -\lambda_{i-1} I(i > 1), & \text{if } j = i - 1, \\ -\lambda_i I(i < n), & \text{if } j = i + 1, \\ 0, & \text{otherwise} \end{cases}$$

and

$$B_{ij} = \begin{cases} -I(i > 1), & j = i - 1, \\ I(i < n), & j = i, \\ 0, & \text{otherwise.} \end{cases}$$

Note that A is a tridiagonal matrix. Denote also $Y = (Y_1, \ldots, Y_n)^T$.

Using this notation and the formula for $\nabla_\mu L(\mu, \rho)$, we can represent the KKT conditions as

$$A \cdot \mu = Y + \frac{1}{2} B \cdot \rho, \tag{4}$$

$$\mu_i \le \mu_{i+1}, \quad i = 1, \ldots, n - 1, \tag{5}$$

$$\rho_i \ge 0, \quad i = 1, \ldots, n - 1, \tag{6}$$

$$\rho_i(\mu_{i+1} - \mu_i) = 0, \quad i = 1, \ldots, n - 1. \tag{7}$$

The algorithm that we introduce here for solving the SMR problem belongs to the class of dual active set algorithms [7,18]. At each iteration, it solves the problem

$$\begin{aligned} \min \; & \sum_{i=1}^{n} (\mu_i - Y_i)^2 + \sum_{i=1}^{n-1} \lambda_i (\mu_{i+1} - \mu_i)^2 \\ \text{s.t.} \; & \mu_i = \mu_{i+1}, \; i \in S, \end{aligned} \tag{8}$$

for a set of active constraints $S \subseteq \{1, 2, \ldots, n - 1\}$. Our algorithm starts with $S = \emptyset$, enlarging it from iteration to iteration, and terminates when the vector μ that solves this problem becomes feasible in the original problem (2).

The correctness of this algorithm is theoretically justified in the accompanying paper [10] which is focused on optimization aspects of our approach. In particular, it is shown in Burdakov and Sysoev [10] that if a constrained optimization problem (8) is solved for some S and the corresponding Lagrange multipliers are all nonnegative, then $\mu_i = \mu_{i+1}$ for all $i \in S$ in (2). Moreover, if the optimal solution to problem (8) is such that $\mu_j \ge \mu_{j+1}$ for some $j \in S' : S' \notin S$, then $\mu_j = \mu_{j+1}$ for $j \in S'$ in (2). These two properties imply that if we set $S = \emptyset$ and compute all $j \in S'$ such that $\mu_j \ge \mu_{j+1}$, these are active constraints in (2) and thus S' can be added to S. Because the constraints corresponding to S' are active in (2), they will also be active in (8), and all Lagrange multipliers in (8) for the current S will be nonnegative. Accordingly, the process of computing of S' and updating S as

$$S \leftarrow S \cup S', S' = \{i \notin S : \; \mu_i \ge \mu_{i+1}\}. \tag{9}$$

can be repeated until S' becomes empty. This will happen when either $\mu_i < \mu_{i+1}$ for all $i \notin S$ or S contains all existing constraints from (2). This implies that the algorithm finds the optimal solution to (2) by doing at most n updates of S.

We will show now how to efficiently solve problem (8). Let $[i, j]$ denote a segment of indexes $\{i, i+1, \ldots, j-1, j\}$. Suppose that $[p, q-1] \subseteq S$. Then the corresponding adjacent active constraints give $\mu_p = \mu_{p+1} = \ldots = \mu_q$. If, in addition, $p - 1 \notin S$ and $q \notin S$, then we call $[p, q]$ a *block*. A singleton set $\{p\} \subseteq [1, n]$ is also called a block if $p - 1 \notin S$ and $p \notin S$. Thus, any active set S generates a partitioning of $[1, n]$ into a set of blocks $\Omega = \{B_1, B_2, \dots, B_{n_B}\}$, where n_B is the number of blocks. Each block $B_i = [p_i, q_i]$ is characterized by its number of elements $n_i = q_i - p_i + 1$, its *observed mean*

$$\bar{Y}_i = \frac{1}{n_i} \sum_{j=p_i}^{q_i} Y_j$$

and its *common block value* $\hat{\mu}_i$ such that

$$\mu_j = \hat{\mu}_i, \quad \forall j \in B_i. \tag{10}$$

Denote $\bar{Y} = (\bar{Y}_1, \dots, \bar{Y}_{n_B})^T$ and $\hat{\mu} = (\hat{\mu}_1, \dots, \hat{\mu}_{n_B})^T$. The blocks are supposed to be sorted in increasing order of the indexes they contain, which means that $p_i - 1 = q_{i-1}$. We denote by $\hat{A}$ a $n_B \times n_B$ tridiagonal matrix whose elements are defined as

$$\hat{A}_{ij} = \begin{cases} 1 + \frac{\lambda_{p_i-1}}{n_i} I(p_i > 1) + \frac{\lambda_{q_i}}{n_i} I(q_i < n), & \text{if } j = i, \\ -\frac{\lambda_{p_i-1}}{n_i} I(p_i > 1), & \text{if } j = i - 1, \\ -\frac{\lambda_{q_i}}{n_i} I(q_i < n), & \text{if } j = i + 1, \\ 0, & \text{otherwise.} \end{cases}$$

The next result shows how to easily compute the solution to problem (8).

Theorem 1 *The common block values of the optimal solution to problem (8) are uniquely defined by the system of linear equations*

$$\hat{A} \cdot \hat{\mu} = \bar{Y}. \tag{11}$$

Proof As it was mentioned above, the solution to problem (8) is uniquely defined by the system of linear Eq. (4), where (10) holds and $\rho_i = 0$ for all $i \notin S$. Consider a block $B_i = [p_i, q_i]$. The corresponding equations in (4) are of the form:

$$\begin{cases} \hat{\mu}_i + \lambda_{j-1}(\hat{\mu}_i - \hat{\mu}_{i-1}) \cdot I(j > 1) = Y_j - \frac{\rho_j}{2} I(j < n) & \text{for } j = p_i \\ \hat{\mu}_i = Y_j + \frac{\rho_{j-1}}{2} I(j > 1) - \frac{\rho_j}{2} I(j < n) & \text{for } j = p_i + 1 \\ \cdots & \\ \hat{\mu}_i + \lambda_j(\hat{\mu}_i - \hat{\mu}_{i+1}) \cdot I(j < n) = Y_j + \frac{\rho_{j-1}}{2} I(j > 1) & \text{for } j = q_i \end{cases}$$

By summing these equations, we obtain

$$n_i \hat{\mu}_i + \lambda_{p_i-1}(\hat{\mu}_i - \hat{\mu}_{i-1}) \cdot I(p_i > 1) + \lambda_{q_i}(\hat{\mu}_i - \hat{\mu}_{i+1}) \cdot I(q_i < n) = \sum_{j=p_i}^{q_i} Y_j$$

A simple algebraic manipulation with this equation results in Eq. (11). □

The important feature of system (11) is that the matrix $\hat{A}$ is tridiagonal. It can be efficiently solved, e.g., by the Thomas algorithm [13] of complexity $O(n)$. Moreover, since $\hat{A}$ does not depend on Y, (11) can be viewed as a linear smoother.

We call the outlined algorithm for solving SMR problem (2) a *smoothed pool adjacent violators* (SPAV) algorithm. It employs Theorem 1.

Algorithm 1 SPAV algorithm

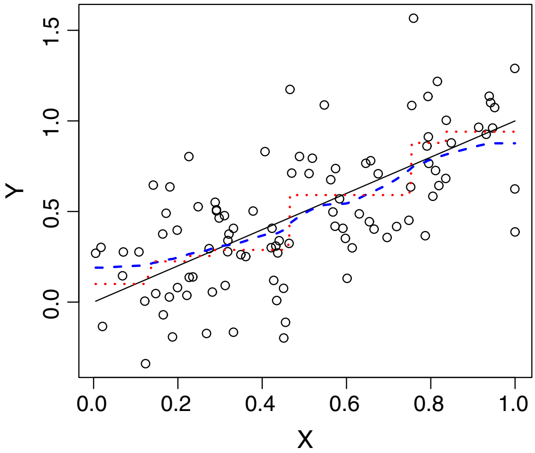

Fig. 1 The solid line is $f(x) = x$; the circles are artificially generated data $Y = f(X) + \epsilon$, where $\epsilon \sim N(0, 0.3)$; the dashed line is the SPAV solution (with linear kernel and λ selected by cross-validation, introduced below); the dotted line is the PAV solution

Given $Y = (Y_1, ...Y_n)$ and $\Lambda = (\lambda_1, \dots, \lambda_{n-1})$.

Set $S = \emptyset$, $n_B = n$, $B_j = \{j\}$ $(\forall j \in [1, n])$ and $\Omega = \{B_1, \dots, B_n\}$.

Smoothing step

1. Compute $\hat{\mu}$ and μ according to (11) and (10), respectively.
2. If $\mu_i \leq \mu_{i+1}$ for all $i = 1, \dots, n-1$ then stop and return μ as the optimal solution, else

PAV step

3. Enlarge S by formula (9), and update Ω and n_B accordingly.
4. Go to step 1.

Step 3 is called PAV step because it pools adjacent violators, i.e., indexes i and $i+1$ satisfying $\mu_i > \mu_{i+1}$ into one set and enforces $\mu_i = \mu_{i+1}$ in the following steps. The result of the SPAV algorithm and the traditional PAV algorithm is compared in Fig. 1. The step-wise response produced by PAV algorithm can sometimes be unrealistic and deviate substantially from the true model. One can see this in Fig. 1, where the SPAV response looks much more natural and closer to the true model.

The following theorem provides an estimate of the complexity of the SPAV algorithm.

Theorem 2 *The complexity of the SPAV algorithm is* $O(n^2)$

Proof Smoothing step followed by PAV step can be viewed as one iteration. As it was mentioned above, the number of iterations does not exceed n. The complexity of step 1 is determined by the complexity of composing and solving a tridiagonal system of equations, which is $O(n)$. Step 2 involves $O(n)$ comparison operations. Step 3 involves a search that requires $O(n)$ operations plus adding an element to a list and updating the list of blocks which requires $O(n)$ steps. Therefore, the total complexity of one iteration is $O(n)$, which means that the total complexity of the algorithm is $O(n^2)$. □

Note that the quadratic complexity is a worst-case estimate, which assumes that the number of iterations is n. In practice, we observed that this number was fewer than n, and the CPU time of SPAV algorithm was growing with n much slower than as n^2.

3 Connections to probabilistic modeling

It is a known fact that penalization of the objective can often be thought of putting a regularization prior on the model, as it is for instance with ridge regression and LASSO [29]. We now consider how the problem (2) can be represented from the probabilistic perspective. Assume the following model for the means:

$$Y_i \sim N(\mu_i, \alpha^2), \tag{12}$$

where the prior on the components of μ are assumed to be of the following form:

$$\mu_{i+1} \sim \mu_i + N_+(0, \sigma_i^2),\ \ i = 1, \dots, n-1, \tag{13}$$

$$p(\mu_1) \propto 1. \tag{14}$$

Here N_+ is the truncated normal distribution that ensures that the sequence of μ_i's is non-decreasing and σ_i^2. The constants σ_i^2 can be arbitrarily chosen but since we assume that the expected response should be a continuous function, it is natural to let σ_i^2 depend on how close X_{i+1} is to X_i: The smaller the distance, the lesser the variance should be; and when $X_{i+1} = X_i$, σ_i^2 should be zero.

We introduce a family of functions K, called here *kernel* (by analogy to the Gaussian Process model), which meet the requirements mentioned above and are defined by

$$\sigma_i^2 = \frac{1}{\beta K\left(X_i, X_{i+1}\right)}, \tag{15}$$

where, for instance,

$$K(X, X') = \frac{1}{|X - X'|^p}. \tag{16}$$

Here β is some constant that steers the level of smoothness: The higher the β, the lesser the variance is and the higher is the degree of smoothness; p is a constant that also steers the flexibility of the model: Higher p allows for larger variation between the nearest points. We shall call the function K, for $p = 1$ and $p = 2$, a *linear kernel* and $p = 2$ *quadratic kernel*, respectively. In principle, one may select some expression for the kernel $K(X, X')$ other than (16); the important is that this function should be non-decreasing with respect to $\left|X - X'\right|$, and $1/K(X, X')$ should become zero for $X = X'$.

An important property of our model is presented in the following result.

Theorem 3 *The solution to problem (2), where*

$$\lambda_i = \frac{\alpha^2}{\sigma_i^2},\quad i = 1, \dots n-1, \tag{17}$$

is the maximum a posteriori estimate provided by model (12)–(14).

Proof According to Bayes theorem, the posterior distribution is

$$p(\mu|Y, X) \propto p(Y|\mu, X) \cdot p(\mu|X).$$

Here, the likelihood is the product of normal distributions

$$p(Y|\mu, X) \propto \exp\left(-\frac{\sum_{i=1}^{n}(Y_i - \mu_i)^2}{2\alpha^2}\right).$$

The prior $p(\mu|X)$ can be computed via the chain rule:

$$p(\mu|X) = p(\mu_1|X) \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i, X).$$

Taking into account that $\mu_i \leq \mu_{i+1}$ for all $i = 1, \ldots, n-1$, we have

$$p(\mu|X) \propto \exp\left(-\frac{\sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2}{2\sigma_i^2}\right) = \exp\left(-1/(2\alpha^2) \sum_{i=1}^{n-1} \lambda_i \, (\mu_{i+1} - \mu_i)^2\right),$$

where λ_i is defined by (17).

By computing the product of the likelihood and the prior, and then taking the logarithm, we get

$$\log p(\mu|Y, X) \propto -\sum_{i=1}^{n} (Y_i - \mu_i)^2 - \sum_{i=1}^{n-1} \lambda_i \, (\mu_{i+1} - \mu_i)^2 ,$$

which implies that the maximum of $p(\mu|Y, X)$ is assured by the solution to problem (2). □

Using this theorem and assuming that the variance is given by (15), we obtain

$$\lambda_i = \frac{\alpha^2}{\sigma_i^2} = \lambda K(X_i, X_{i+1}), \tag{18}$$

where $\lambda = \alpha^2 \cdot \beta$. Given that the noise variance α^2 is fixed, the smoothing parameter λ plays the same role as β: The higher λ is, the greater is the degree of smoothness of the fitted SPAV response.

For the kernel defined by (16) with $p = 2$, formula (18) indicates that the penalty terms in (2) takes the form

$$\lambda_i (\mu_{i+1} - \mu_i)^2 = \lambda \left(\frac{\mu_{i+1} - \mu_i}{X_{i+1} - X_i}\right)^2 .$$

Then, in this case, smoothing is performed by penalizing the squared slope of the expected response where the slope is computed by using a finite-difference approximation.

Our analysis enables us to conclude that, using the probabilistic approach, it is possible to select an appropriate structure of λ_i in such a way that it reflects the location of the X components. Without using probabilistic modeling, this structure would perhaps be difficult to determine. Another advantage of using probabilistic methodology is that it provides a strategy for determining consistent predictions for some new values of X, as it is discussed below.

Prediction for monotonic regression is a special problem, as it is discussed in Sysoev et al. [41].

In general, the MR is only able to compute unique predictions for the values of the input levels that are present in the training data. This can be a critical obstacle for applying MR in many machine learning applications. However, for the smoothed MR, we can define a consistent prediction model for some new predictor value X^* as follows.

Consider the following model for the expected response μ^* that corresponds to a predictor value X^* which is located between X_i and X_{i+1}:

$$\mu^*|\mu_i \sim \mu_i + N_+(0, \sigma_{i,*}^2), \tag{19}$$

$$\mu_{i+1}|\mu^* \sim \mu^* + N_+(0, \sigma_{*,i+1}^2), \tag{20}$$

where the σ values are defined by (15) in which variables (X_i, X_{i+1}) are substituted by (X_i, X^*) or (X^*, X_{i+1}), respectively. If X^* is outside the observed interval, i.e., $X^* > X_n$ or $X^* < X_1$, then we simply define the predicted value for μ^* as μ_n or μ_1, respectively.

The prediction model presented by (19) and (20) is a natural extension of the model given by (13) because the two models have the same structure. Accordingly, we can derive the prediction in the same way as it was done for (13), i.e., by computing a maximum a posteriori estimate of $\mu^*|\mu_i, \mu_{i+1}$.

Theorem 4 *Given model (19), (20), the maximum a posteriori estimate of $\mu^*|\mu_i, \mu_{i+1}$ can be computed as*

$$\mu^* = \frac{K(X_i, X^*)\mu_i + K(X^*, X_{i+1})\mu_{i+1}}{K(X_i, X^*) + K(X^*, X_{i+1})}. \tag{21}$$

Proof By Bayes formula and the chain rule, we get:

$$p(\mu^*|\mu_i, \mu_{i+1}) = \frac{p(\mu^*, \mu_i, \mu_{i+1})}{p(\mu_i, \mu_{i+1})} = \frac{p(\mu_i)}{p(\mu_i, \mu_{i+1})} p(\mu^*|\mu_i) p(\mu_{i+1}|\mu^*).$$

Therefore,

$$\log p(\mu^*|\mu_i, \mu_{i+1}) \propto -\frac{(\mu^* - \mu_i)^2}{2\sigma_{i,*}^2} - \frac{(\mu_{i+1} - \mu^*)^2}{2\sigma_{*,i+1}^2}.$$

By maximizing the expression above and using (15), we get

$$2\beta K(X_i, X^*)(\mu^* - \mu_i) + 2\beta K(X^*, X_{i+1})(\mu^* - \mu_{i+1}) = 0,$$

which leads to (21). □

From the practical machine learning perspective, it is crucial that instead of optimizing $n - 1$ parameters in (2), we need to estimate only one parameter λ. A possible way of doing this is cross-validation. The standard cross-validation implies that several λ values need to be considered, and the SMR needs to be computed for each λ value. This leads to the overall computational time $O(n^2)$ (because the number of considered λ values is normally smaller than n and is not dependent on n). However, applying the cross-validation may still be computationally expensive.

We suggest a procedure called here *generalized cross-validation*. It is based on the following idea supported by empirical observations. The expected (true) response should normally be increasing, and there is a little chance of observing a region where the response is constant. Since in our model the response is constant for the observations that were merged into one block by the PAV step of the SPAV algorithm, we can conclude that in an ideal model there should be very few executions of the PAV step. Accordingly, if we remove the PAV steps from this algorithm, it should not affect much this ideal model, although the monotonicity may be violated. The algorithm can be presented as follows.

Algorithm 2 Smoothing algorithm

```
Given $Y = (Y_1, ...Y_n)$ and $\Lambda = (\lambda_1, \dots, \lambda_{n-1})$.
    Solve (4).
```

The fitted responses produced by the SPAV and Smoothing algorithms are presented in Fig. 2, which demonstrates that these responses are quite similar.

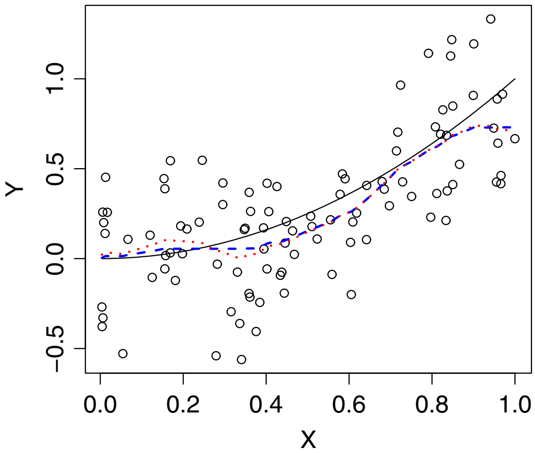

Fig. 2 An artificially generated data $Y = f(X) + \epsilon, \epsilon \sim N(0, 0.3)$ where the solid line is $f(x) = x^2$, the dashed line is the response obtained by the SPAV algorithm (cross-validated, linear kernel) and Smoothing algorithm with the same λ as in the SPAV algorithm (dotted line)

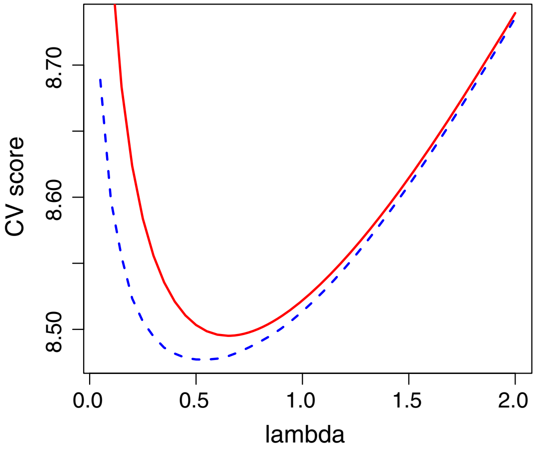

Fig. 3 Cross-validation curve (solid) and the generalized cross-validation scores (dashed) for some artificially generated data, $n = 100$

We define the generalized cross-validation as a standard cross-validation in which Algorithm 2 is applied instead of Algorithm 1 in order to determine an optimal penalty λ. This strategy allows for substantially reducing the computational time. Our numerical experiments confirm that the local optimum of the cross-validation curve and the generalized cross-validation curves are typically located close to each other, as it is illustrated in Fig. 3.

4 Correction on the boundaries

The MR is known to be inconsistent on the boundaries, which means that the MR estimates for μ_1 and μ_n do not converge to the true values when n tends to infinity, and some solutions to this problem were suggested [34,43]. This problem is also known as the spiking problem. An intuitive explanation of this issue is the asymmetry of the PAV algorithm on the boundary: The observation with index n can only be merged into a block with the observations $i < n$, while the observation with index j located in the middle of the domain of X can be merged into a block with neighbors $i > j$ and $i < j$. It may happen that $Y_n > Y_i$ for all $i < n$. In this case, the PAV algorithm does not make any error correction and returns $\hat{\mu}_n = Y_n$.

A similar kind of asymmetry can be observed in the SMR problem: The objective function in (2) contains two penalty factors (involving $i+1$ and $i-1$) for all $i = 2, \ldots, n-1$, while for $i = n$ or $i = 1$ there is only one penalty factor. Figure 1 demonstrates that the SPAV fitted response can sometimes behave badly on the boundaries, i.e., it can be shifted toward the observed values of Y_1 and Y_n.

In order to resolve this issue, we apply the idea presented in Wu et al. [43]. More specifically, we add two extra terms aimed at penalizing too high values of μ_n and too low values of μ_1:

$$\begin{array}{ll} \min \sum_{i=1}^{n} (\mu_i - Y_i)^2 + \sum_{i=1}^{n-1} \lambda_i (\mu_{i+1} - \mu_i)^2 + \phi\mu_n - \phi\mu_1 \\ \text{s.t.} \qquad\qquad\qquad \mu_i \le \mu_{i+1}, \; i = 1, \ldots, n-1, \end{array} \tag{22}$$

where $\phi > 0$ is a penalty factor.

It can be easily verified that

$$(\mu_1 - Y_1)^2 + (\mu_n - Y_n)^2 + \phi\mu_n - \phi\mu_1 = \left(\mu_1 - Y_1 - \frac{\phi}{2}\right)^2 + \left(\mu_n - Y_n + \frac{\phi}{2}\right)^2 + c,$$

where c does not depend on μ_1 or μ_n. Like in Wu et al. [43], this observation allows for reducing problem (22) to (2) by shifting the first and the last observed response values as follows:

$$Y_1 \leftarrow Y_1 + \frac{\phi}{2} \quad \text{and} \quad Y_n \leftarrow Y_n - \frac{\phi}{2}.$$

Then problem (2) modified in this way can be solved by means of the SPAV algorithm. The only requirement is to use the values $Y' = Y + \phi \cdot e$ instead of Y, where $e = (1/2, 0, \ldots, 0, -1/2)^T$.

Consider now how to choose the parameter ϕ. One possibility is to use cross-validation, as it was suggested for λ. However, selecting an optimal combination of two parameters might be computationally expensive, and we describe an alternative approach that appeared to work well in practice.

The SPAV algorithm solves problem (11) repeatedly, and if Y is replaced by Y', the equation becomes

$$\hat{A} \cdot \hat{\mu} = \bar{Y}' = \bar{Y} + \phi\bar{e}, \tag{23}$$

that is

$$\hat{\mu} = \hat{A}^{-1}\bar{Y} + \phi\hat{A}^{-1}\bar{e}, \tag{24}$$

where $\bar{e} = \left(\frac{1}{2n_1}, 0, \ldots, 0, \frac{-1}{2n_m}\right)$, and n_1 and n_m are sizes of blocks 1 and $m = n_B$, respectively. We suggest to select ϕ in such a way that $\hat{\mu} = \hat{\mu}(\phi)$ approximates the observed data $\bar{Y}$ in the best way in the least-squares sense:

$$\min_{\phi} (\hat{\mu}(\phi) - \bar{Y})^T (\hat{\mu}(\phi) - \bar{Y}).$$

This yields the optimal estimate

$$\hat{\phi} = \frac{(\bar{Y} - \mu')^T e'}{\|e'\|^2},$$

where $\mu' = \hat{A}^{-1}\bar{Y}$ and $e' = \hat{A}^{-1}\bar{e}$. Therefore, the fitted value $\hat{\mu}$ at each Smoothing Step of the SPAV algorithm can be computed as follows:

Algorithm 3 Step 1 of SPAV with correction

```
1. Solve $\hat{A}\mu' = \bar{Y}$
2. Solve $\hat{A}e' = \bar{e}$
3. Compute $\hat{\phi} = \frac{(\bar{Y}-\mu')^T e'}{\|e'\|^2}$
4. Compute $\hat{\mu} = \mu' + \hat{\phi} \cdot e'$
```

Note that the linear equations in Algorithm 3 involve tridiagonal matrices only, and the rest of the algorithm contains simple vector operations. Therefore, the complexity of computing the correction is $O(n)$, and the overall $O(n^2)$ complexity of the SPAV algorithm is not affected by introducing the correction.

5 Numerical results

Modern machine learning methods are often required to process large and complex data, and this needs to be taken into account when examining the performance of the fitting methods. In order to evaluate the efficiency of the proposed smoothing approach, we compared it to the following methods:

- monotonic smoothers m_{IS} and m_{SI} described in Mammen [24],
- PAV solution m_{PAV},

and to two more recent methods:

- shape constrained additive models (SCAM), described in Pya and Wood [35],
- Bayesian isotonic regression (BIR), introduced in Neelon and Dunson [30] and Neelon and Dunson [31].

The former two methods have a fixed complexity estimate, which is $O(n^2)$, while the performance of the two latter methods depends on the number of knots.

Intuition dictates that the number of knots in a model should be dependent on the complexity of the underlying data generating process, and it was confirmed by our preliminary simulations. We considered data sets for various values of n and response $Y = X + \sin(X) + \epsilon$, where $X \sim U[0, \lfloor n/3 \rfloor]$ and $\epsilon \sim N(0, 1)$. It is obvious that both the size and complexity of the data increase with n. It appeared that in order to capture the features of the underlying data, it was necessary for the BIR and SCAM methods to increase the number of knots in proportion to n. Figure 4 represents one such case.

In order to study how BIR and SCAM methods perform in case of large and complex data, i.e., when both size and complexity of the data increase, we performed simulations where data were generated according to the model above, and the number of knots was selected as $k = n/2$ for SCAM and $k = n/5$ for BIR, and each model was generated 100 times. Tables 1 and 2 demonstrate the CPU time averaged over model instances. By applying a polynomial regression to these results, it can be concluded that the complexity grows in practice faster than quadratically with n for these approaches. By extrapolating the results, it can be concluded that for $n = 100{,}000$ observations and the settings described above the SCAM algorithm will take approximately 6 days while the BIR algorithm will require approximately 220 days. Accordingly, it can be prohibitively expensive to fit large and complex data by these methods if a high quality of prediction is required. In contrast, the computational complexity of the

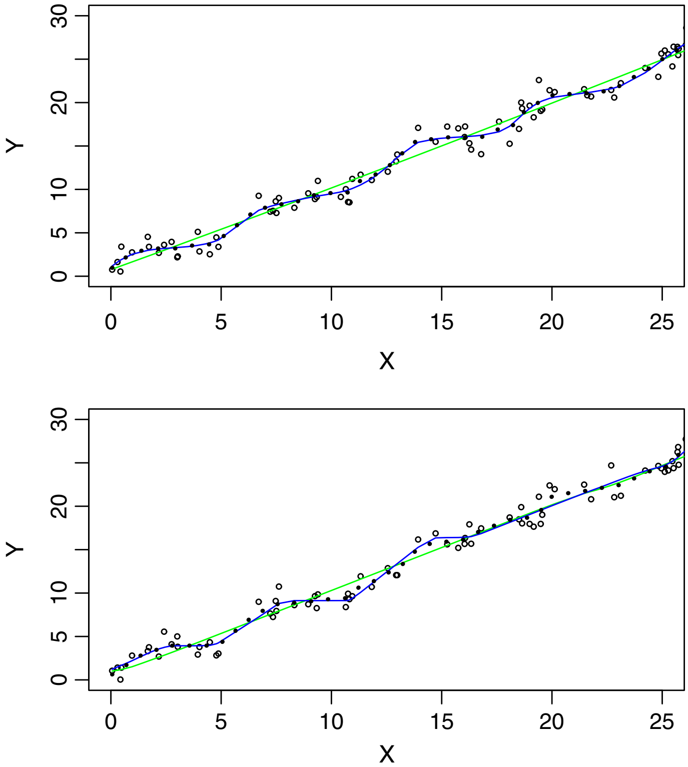

Fig. 4 Data were generated as $X \sim U[0, 50]$, $Y = X + \sin(X) + \epsilon$, $\epsilon \sim N(0, 1)$, N = 150. The top panel: data are fit by SPAV (dotted line), SCAM with 40 knots (light curve) and SCAM with 80 knots (dark curve). The bottom panel: data are fit by SPAV (dotted line), BIR with 5 knots (light curve) and BIR with 30 knots (dark curve). Both figures are zoomed to $X \in [0, 25]$. Both panels demonstrate a clear underfitting of SCAM and BIR when the amount of knots is not sufficient

Table 1 Computational time (s) for SCAM algorithm with a number of knots increasing with n

n	100	200	300	400	500	600	700	800	900	1000
Time	0.28	1.25	4.07	8.85	16.7	18.2	30.1	39.4	50.5	77.6

Table 2 Computational time (s) for BIR algorithm with a number of knots increasing with n

n	100	120	140	160	180	200	220	240	260	280	300
Time	12.2	17.6	24.0	31.2	39.9	49.5	60.3	73.1	86.7	101.6	118.7

SPAV method is only dependent on the amount of observations, and its practical complexity grows almost linearly with n [10]. For the aforementioned model with $n = 100{,}000$, the SPAV algorithm takes less than a minute.

Our comparative simulation studies involve the data generated as $X \sim U[0, A]$, $Y = f(X) + \epsilon$, $\epsilon \sim N(0, s)$ for all combinations of the following settings: $n = 100$, 1000 or 10,000; $s = 0.03$ or 0.1, the functional shape f is one of the following functions $f_1(X) =$

X, $f_2(X) = X^2$, $f_3(X) = (X + \sin(X))/10$, $f_4(X) = \tanh\left(\frac{n}{10}(X - 0.5)\right)$, and $A = \lfloor n/5 \rfloor$ when f_3 is used and $A = 1$ otherwise. These functions were chosen in order to represent various possible behaviors of the underlying model such as linearity, nonlinearity, increasing complexity or a sudden level shift. The latter property is dismissed by many methods although the change points are often present in machine learning applications. To measure the uncertainty of our results, we generate $M = 100$ instances for the same model.

Each data set generated in our simulations was processed by:

- m_1: SPAV algorithm with linear kernel with correction where λ was selected by the generalized cross-validation with 10 folds,
- m_2: SPAV algorithm with linear kernel with correction where λ was selected by the cross-validation with 10 folds,
- m_3: SPAV algorithm with quadratic kernel without correction where λ was selected by the generalized cross-validation with 10 folds,
- m_4: SPAV algorithm with linear kernel without correction where λ was selected by the generalized cross-validation with 10 folds.
- m_{SCAM}: SCAM algorithm. Due to time limitations, we fixed $k = 15$ as it was done in the R package documentation of SCAM.
- m_{BIR}: BIR algorithm. Due to time limitations, we fixed $k = 3$ as it was recommended in Neelon and Dunson [30].
- monotonic smoothers m_{IS} and m_{SI},
- monotonic regression m_{PAV},

The mean squared error (which is a measure of the predictive performance)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(m_*(X_i) - f(X_i))^2$$

is computed, where m_* is a fitted response computed by some method. The average MSE values computed for different settings are presented in Table 3. Standard errors $se\left(\overline{\mathrm{MSE}}\right)$ for each average MSE value $\overline{\mathrm{MSE}}$ were computed from the set of the MSE values $\{\mathrm{MSE}_1, \ldots, \mathrm{MSE}_M\}$ as

$$\mathrm{se}\left(\overline{\mathrm{MSE}}\right) = \frac{1}{M}\sqrt{\sum_{i=1}^{M}(\mathrm{MSE}_i - \overline{\mathrm{MSE}})^2},$$

where M is the amount of instances generated per model. The standard errors are given in Table 4.

By analyzing MSE values, it can be concluded that the SPAV algorithm has two main competitors: the SCAM algorithm and the PAV algorithm. The other tested algorithms appear to have a worse predictive performance than one of these three algorithms.

When the underlying model is simple, i.e., when it is given by functions f_1 or f_2, the average MSE of the SCAM algorithm is smaller than the average MSE values of all other algorithms, including SPAV. However, it must be noted that the SPAV algorithm is ranked second for these cases, and its MSE values are often quite close to those of the SCAM algorithm. Moreover, for some settings the difference is not statistically significant.

When the underlying model is complex or has sharp level shifts (change points), the average MSE values of the SCAM algorithm are quite high even for small n values, and the predictive performance dramatically decreases for large n values. For such underlying models, the SPAV algorithm is significantly more precise than the SCAM algorithm. Our results

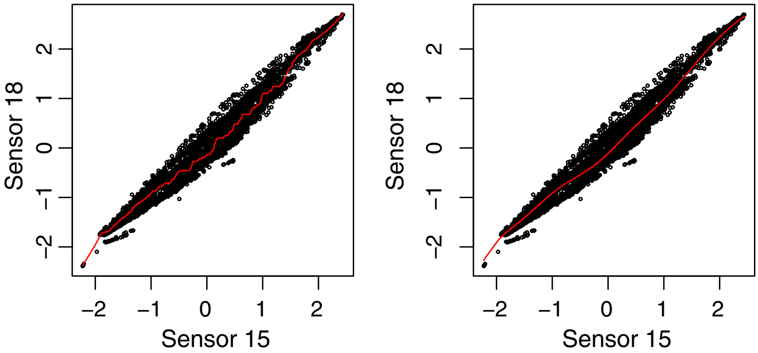

Fig. 5 A sample of $n = 10^4$ observations from the 'Gas sensor' data fitted by m_1 model (left panel) and by m_{SCAM} model (right panel)

demonstrate that the SPAV algorithm has the lowest MSE values for these underlying models, while the PAV algorithm is ranked second, and the difference in predictive performance between these two algorithms is statistically significant.

By comparing the variants of the SPAV algorithm in Table 3, it can be concluded that applying the boundary correction often helps to improve the predictive power of the algorithm, and the effect is quite substantial for simpler models. It appears that applying generalized cross-validation instead of the usual cross-validation often leads to similar MSE values, which was expected due to the results presented in Sect. 3. Results presented in the table also indicate that employing a linear kernel seems to lead to a better predictive performance than using a quadratic kernel.

In order to further investigate the performance of the SPAV approach, we consider two real datasets: 'Gas sensor' data [15] (containing around 4 million observations) and 'Geo-Magnetic field' data [6] (containing around fifty thousand observations). The datasets were downloaded from UCI Machine Learning repository [3]. Figures 5 and 6 demonstrate measurements of a pair of sensors for samples of $n = 10^4$ observations extracted from these datasets. Both figures lead to the conclusion that relationship between the corresponding sensors is reasonable to model as a monotonic function. We fit models m_1 and m_{SCAM} (which appear to be the main competitors in our simulation studies) to these samples by taking 'Sensor 15' as X variable and 'Sensor 18' as Y variable in the 'Gas sensor' data and 'Acceleration X sensor' as X variable and 'Y-axis angle roll sensor' as Y variable in 'Geo-Magnetic field' data. The predictions from the fitted models are shown in Figures 5 and 6 as solid lines.

To make a comparison of the predictive performance of m_1 and m_{SCAM}, we apply the following procedure to each of the two datasets. In step 1, we sample (non-overlapping) training and test data of the size $n = 10^4$ from the given dataset. In step 2, we fit m_1 and m_{SCAM} to the training data (in model m_1, the generalized cross-validation is applied to the current training data) and compute the MSE values for the test data. Steps 1 and 2 are repeated $M = 100$ times, and the obtained set of the MSE values is used to compute the mean MSE value and the standard error of the mean MSE for both models. Table 5 illustrates the results. In addition, we compute the test errors by sampling $n = 10^5$ observations from the 'Gas sensor' data and fitting m_1 and m_{SPAV} models. The test error for m_1 is 29.09×10^{-3} while for m_{SCAM} it is 30.85×10^{-3}.

It can be concluded from the presented analysis of real datasets that m_1 has a statistically significantly better performance than m_{SCAM} for these data. A possible explanation for this

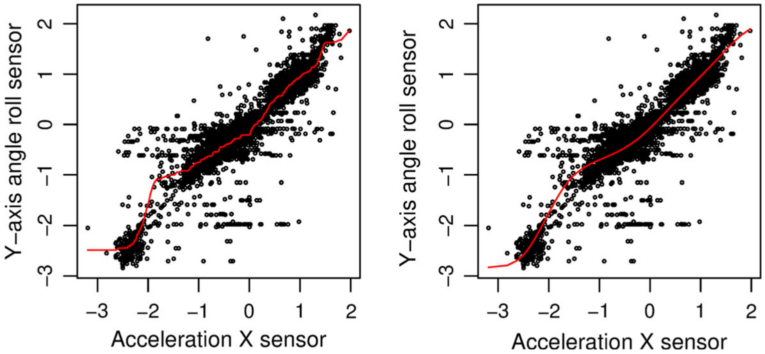

Fig. 6 A sample of $n = 10^4$ observations from the 'Geo-magnetic field' data fitted by m_1 model (left panel) and by m_{SCAM} model (right panel)

result can be that m_1 model is flexible enough to discover small but significant trends in the data which are not discovered by m_{SCAM}(as Fig. 5 illustrates), and m_1 is better than m_{SCAM} in modeling sharp level shifts (as Fig. 6 illustrates).

6 Conclusions

In this work, we introduce an approach for smoothed monotonic regression in one predictor variable. It allows for adjusting the degree of smoothness to the data. We have shown that the worst-case computational complexity of this method is $O(n^2)$ which makes the method suitable in large-scale settings. Moreover, it is found in the accompanying paper [10] that, in practice, its running time grows almost linearly with n for a given set of penalty factors Λ.

In many machine learning applications, the dependence between the response and predictor variables is a complicated function. Our numerical experiments demonstrate that the predictive performance of SCAM and BIR methods can substantially degrade when the complicated data are involved, unless a sufficiently large amount of knots is used. At the same time, it may be impossible to choose a proper number of knots in these algorithms without making them prohibitively too expensive. The SMR method is free of this shortcoming. Our simulations demonstrate that, in these settings, the SMR method has the best predictive performance compared to the other tested algorithms. It should be emphasized that the computational cost of the our approach does not depend on the complexity of the underlying response function.

Another obvious advantage of the SMR method is that it is able to treat both smooth or non-smooth responses: For small values of the smoothing parameter, a response close to piecewise-constant is produced, and the increasing smoothing parameter value increases the degree of smoothness. Our numerical experiments reveal that our approach has a superior predictive performance in situations when there is a sharp change in the level in the response function, and that is important for the applications characterized by the presence of change points.

The SMR problem was introduced as a regularization problem with n regularization parameters. Then, a connection to a probabilistic model was established which assured the same MAP estimate as the solution to the regularization problem. The probabilistic formulation provided us with a clear strategy for reducing n regularization parameters to just one parameter by properly setting the structure of the prior variance and thus reducing the overall worst-case computational cost to $O(n^2)$. The prior variance depends on the kernel structure, and the predictive performance was better when linear kernels were used compared to the case with quadratic kernels. In addition, the probabilistic formulation provided us with a consistent predictive model for the input values which are not present in the training data. Further, we motivated a generalized cross-validation strategy for the selection of the smoothing parameter. Our preliminary experiments confirmed that the generalized cross-validation produces results which are similar to the cross-validation, and these required far less computational efforts. Finally, a boundary effect problem was studied, and a strategy for the inconsistency correction was suggested. The proposed strategy had a clear positive effect on the predictive performance of the SMR method.

For estimating the predictive performance of the SMR method, it was natural to use MSE values (as we have done in our numerical experiments). For practical data, measures of uncertainty, such as confidence (credible) limits, can also be derived. A natural way of deriving such intervals would be to use the probabilistic formulation (12) and derive the posterior distribution of the quantity of interest, but this would require a full Bayesian inference for this model. This is avoided in our scalable algorithms. An alternative (which we recommend) is to use bootstrap techniques [14] together with the estimator provided by the result of Algorithms 1 or 3.

Some peculiarities of our model are noteworthy. First, although the fitted response is continuous, it is in general not differentiable. This is because the predictive model uses only two adjacent fitted values to make prediction for a new input value, and this may result in that the left and right derivatives of the fitted response may be different in some points. Another peculiarity is that the fitted response and the prediction depend on the choice of the kernel, but this is also typical for many other methods, such as kernel smoothers.

The SMR problem was formulated for the case of one predictor variable, and one can directly use our model in a generalized additive model setting to fit the models with many predictors. Another possible way of fitting multivariate data could be to generalize our frequentist model as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} (\mu_i - Y_i)^2 + \sum_{j,k} \lambda_{jk} \left(\mu_j - \mu_k\right)^2 \\ \text{s.t. } & \mu_j \le \mu_k, \ \text{iff } X_j \preceq X_k \ \forall j, k = 1, \ldots, n \end{aligned}$$

For small n, the corresponding quadratic programming problem can be solved by using conventional optimization algorithms, but for large n, these algorithms are prohibitively expensive. Finding efficient ways of solving this problem is a direction of our future research. Another interesting direction is to state a joint fitted and predictive model and then perform its full Bayesian analysis.

7 Appendix

Table 3 Average MSEs times 10^5 of the given algorithms measured for given standard deviations of the error s, sample sizes n and functional forms $f = f_i$, where $i = 1, \ldots 4$. Averages were computed over 100 instances. Two best algorithms are selected in bold (SPAV implementation versions are treated as one algorithm)

s	f	n	m_1	m_2	m_3	m_4	m_{SCAM}	m_{BIR}	m_{IS}	m_{SI}	m_{PAV}
0.03	1	100	**2.91**	3.13	15	11.63	**2.22**	11.35	16.73	23.08	29.45
0.1	1	100	**31.12**	31.35	80.97	74.17	**25.72**	38.61	88.76	100.79	158.94
0.03	2	100	**11.06**	11.35	15.28	12.5	**6.22**	35.45	19.23	26.24	28.23
0.1	2	100	**76.76**	78.89	86.49	79.33	**55.16**	83.18	93.47	111.69	164.56
0.03	3	100	**19.73***	21.25	25.57	22.94	153.26	445.36	53.02	53.27	**39.57**
0.1	3	100	**129.07***	138.19	152.02	139.46	356.47	477.51	**154.61**	207.51	231.57
0.03	4	100	16.72	18.28	20.9	**15.8**	**9.86**	596.79	61.4	69.56	29.38
0.1	4	100	124.72	131.61	141.44	**116.62**	**54.58**	676.87	148.12	164.82	211.05
0.03	1	1000	**0.30**	0.32	2.47	1.99	**0.25**	0.34	2.63	2.99	6.66
0.1	1	1000	3.65	**3.45**	13.31	12.12	**3.03**	4.07	18.80	17.17	34.61
0.03	2	1000	**1.87**	1.89	2.71	2.16	**0.86**	29.17	2.86	3.46	6.22
0.1	2	1000	**12.42**	12.64	14.74	13.53	**7.21**	27.01	18.41	19.26	34.28
0.03	3	1000	**18.22**	18.84	22.31	18.36	499.78	499.32	588.12	617.33	**37.65**
0.1	3	1000	114.6	116.58	129.38	**114.57**	503.21	503.77	614.29	618.64	**219.01**
0.03	4	1000	3.24	**2.33**	75.70	3.22	4946.93	4482.64	133.93	131.19	**3.41**
0.1	4	1000	29.49	**18.57**	90.09	28.02	4860.21	5060.60	156.09	146.52	**25.5**
0.03	1	10000	**0.05**	0.05	0.38	0.33	**0.02**	0.03	0.59	0.62	1.38
0.1	1	10000	**0.54**	0.56	2.11	2.02	**0.25**	0.39	2.62	2.93	7.03
0.03	2	10000	**0.34**	0.34	0.43	0.39	**0.17**	33.58	0.91	0.94	1.33
0.1	2	10000	**2.04**	2.04	2.43	2.31	**0.89**	37.70	2.86	3.29	6.92
0.03	3	10000	**18.23**	18.8	22.23	18.23	500.04	500.03	15278.99	15293.08	**37.94**
0.1	3	10000	**115.24**	116.41	130.44	115.35	500	500.11	15295.9	15289.35	**222.58**
0.03	4	10000	1.56	**0.27**	49.12	1.54	6486.98	2615.02	855.61	853.56	**0.38**
0.1	4	10000	2.41	**2.31**	50.54	2.41	6496.67	2808.87	845.26	838.8	**3.06**

Table 4 Standard errors of the average MSEs times 10^5 of the given algorithms measured for given standard deviations of the error s, sample sizes n and functional forms $f = f_i$, where $i = 1, \ldots 4$. Averages were computed over 100 instances

s	f	n	m_1	m_2	m_3	m_4	m_{SCAM}	m_{BIR}	m_{IS}	m_{SI}	m_{PAV}
0.03	1	100	0.390	0.462	0.448	0.414	0.232	0.749	0.512	1.126	0.553
0.1	1	100	3.699	3.624	4.186	4.470	2.692	3.231	3.823	4.557	5.027
0.03	2	100	0.365	0.389	0.376	0.382	0.270	2.239	0.580	1.727	0.534
0.1	2	100	4.003	4.276	3.729	3.805	3.622	4.071	3.864	4.745	4.805
0.03	3	100	0.540	0.669	0.539	0.578	12.632	4.898	2.369	5.461	0.646
0.1	3	100	5.160	5.530	4.777	4.727	15.449	6.139	5.043	8.783	5.912
0.03	4	100	0.601	0.624	0.586	0.551	0.585	38.384	5.099	6.706	0.665
0.1	4	100	5.284	5.408	5.427	5.066	3.629	29.765	6.304	7.378	5.632
0.03	1	1000	0.035	0.037	0.063	0.059	0.027	0.031	0.062	0.070	0.095
0.1	1	1000	0.421	0.382	0.496	0.466	0.276	0.342	0.508	0.525	0.624
0.03	2	1000	0.046	0.047	0.057	0.052	0.035	2.811	0.057	0.093	0.074
0.1	2	1000	0.483	0.506	0.463	0.465	0.363	2.281	0.496	0.609	0.672
0.03	3	1000	0.145	0.148	0.158	0.143	1.146	1.146	5.092	32.392	0.212
0.1	3	1000	1.288	1.351	1.327	1.281	1.218	1.243	4.610	7.438	1.760
0.03	4	1000	0.109	0.072	2.735	0.113	35.331	150.595	5.425	4.893	0.082
0.1	4	1000	0.978	0.688	2.990	0.928	37.102	285.534	5.494	5.374	0.703
0.03	1	10000	0.003	0.003	0.007	0.007	0.002	0.003	0.008	0.009	0.011
0.1	1	10000	0.027	0.032	0.058	0.057	0.023	0.028	0.057	0.065	0.086
0.03	2	10000	0.009	0.008	0.009	0.008	0.007	3.185	0.013	0.014	0.011
0.1	2	10000	0.056	0.056	0.057	0.058	0.034	3.875	0.062	0.069	0.088
0.03	3	10000	0.051	0.054	0.052	0.051	0.368	0.369	89.292	89.846	0.076
0.1	3	10000	0.403	0.414	0.438	0.401	0.354	0.353	95.412	94.272	0.522
0.03	4	10000	0.026	0.008	7.020	0.026	16.191	575.552	7.724	7.703	0.009
0.1	4	10000	0.225	0.095	7.119	0.219	16.003	473.979	8.049	8.031	0.088

Table 5 Mean test MSE × 10^{-3} values computed for 'Gas sensor' and 'Geo-Magnetic field' data; the standard error values are shown after '±' sign

	m_1	m_{SCAM}
'Gas sensor' data	29.21 ± 0.05	30.88 ± 0.05
'Geo-Magnetic field' data	90.3 ± 0.4	93.8 ± 0.4

References

1. Acton S, Bovik A (1998) Nonlinear image estimation using piecewise and local image models. IEEE Trans Image Process 7:979–991
2. Ant-Sahalia Y, Duarte J (2003) Nonparametric option pricing under shape restrictions. J Econ 116:9–47
3. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html
4. Ayer M, Brunk HD, Ewing GM, Reid W, Silverman E (1955) An empirical distribution function for sampling with incomplete information. Ann Math Stat 26(4):641–647
5. Barlow R, Bartholomew D, Bremner J, Brunk H (1972) Statistical inference under order restrictions. Wiley, New York

6. Barsocchi P, Crivello A, La Rosa D, Palumbo F (2016) A multisource and multivariate dataset for indoor localization methods based on WLAN and geo-magnetic field fingerprinting. In: International Conference on indoor positioning and indoor navigation (IPIN), 2016. IEEE, pp 1–8
7. Best M (1984) Equivalence of some quadratic programming algorithms. Math Program 30(1):71–87
8. Bollaerts K, Eilers P, Mechelen I (2006) Simple and multiple P-splines regression with shape constraints. Br J Math Stat Psychol 59(2):451–469
9. Brezger A, Steiner W (2008) Monotonic regression based on bayesian P-splines. J Bus Econ Stat 26(1):90–104
10. Burdakov O, Sysoev O (2017) A dual active-set algorithm for regularized monotonic regression. J Optim Theory Appl 172(3):929–949
11. Burdakov O, Sysoev O, Grimvall A, Hussian M (2006b) An $o(n^2)$ algorithm for isotonic regression problems. In: Di Pillo G, Roma M (eds) Nonconvex optimization and its applications. Springer, Berlin, pp 25–33
12. Chen Y, Samworth R (2015) Generalized additive and index models with shape constraints. J R Stat Soc Ser B (Stat Methodol) 78:729–754
13. de Boor C (1972) Elementary numerical analysis. McGraw-Hill, New York
14. Efron B, Tibshirani R (1994) An introduction to the bootstrap. CRC Press, Boca Raton
15. Fonollosa J, Sheik S, Huerta R, Marco S (2015) Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. Sens Actuators Chem 215:618–629
16. Friedman J, Hastie T, Tibshirani R (2001) The elements of statistical learning, Springer series in statistics, vol 1. Springer, Berlin
17. Friedman J, Tibshirani R (1984) The monotone smoothing of scatterplots. Technometrics 26(3):243–250
18. Goldfarb D, Idnani A (1983) A numerically stable dual method for solving strictly convex quadratic programs. Math Program 27(1):1–33
19. Hall P, Huang L (2001) Nonparametric kernel regression subject to monotonicity constraints. Ann Stat 29:624–647
20. Holmes C, Heard N (2003) Generalized monotonic regression using random change points. Stat Med 22(4):623–638
21. Kalish ML, Dunn J, Burdakov O, Sysoev O (2016) A statistical test of the equality of latent orders. J Math Psychol 70:1–11
22. Lee C (1996) On estimation for monotone dose-response curves. J Am Stat Assoc 91:1110–1119
23. Lin L, Dunson D (2014) Bayesian monotone regression using gaussian process projection. Biometrika 101(2):303–317
24. Mammen E (1991) Estimating a smooth monotone regression function. Ann Stat 19:724–740
25. Maxwell WL, Muckstadt JA (1985) Establishing consistent and realistic reorder intervals in production–distribution systems. Oper Res 33(6):1316–1341
26. Meyer M (2008) Inference using shape-restricted regression splines. Ann Appl Stat 2:1013–1033
27. Meyer M (2012) Constrained penalized splines. Can J Stat 40(1):190–206
28. Mukerjee H (1988) Monotone nonparametric regression. Ann Stat 16:741–750
29. Murphy K (2012) Machine learning: a probabilistic perspective. MIT Press, Cambridge
30. Neelon B, Dunson DB (2003) Bayesian isotonic regression and trend analysis. Technical report, University of North Carolina, National Institute of Environmental Health Sciences. http://ftp.stat.duke.edu/WorkingPapers/03-15.html. Accessed 11 Sept 2016
31. Neelon B, Dunson DB (2004) Bayesian isotonic regression and trend analysis. Biometrics 60(2):398–406
32. Neelon B, Dunson DB R code for ‘Bayesian isotonic regression and trend analysis’. http://people.musc.edu/~brn200/. Accessed 11 Sept 2016
33. Nocedal J, Wright S (2006) Numerical optimization. Springer, Berlin
34. Pal J (2008) Spiking problem in monotone regression: penalized residual sum of squares. Stat Probab Lett 78(12):1548–1556
35. Pya N, Wood SN (2015) Shape constrained additive models. Stat Comput 25(3):543–559
36. Ramsay JO (1988) Monotone regression splines in action. Stat Sci 3:425–441
37. Riihimäki J, Vehtari A (2010) Gaussian processes with monotonicity information. AISTATS 9:645–652
38. Robertson T, Wright F, Dykstra R (1988) Order restricted statistical inference. Wiley, New York
39. Shively T, Sager T, Walker S (2008) A bayesian approach to non-parametric monotone function estimation. J R Stat Soc Ser B (Stat Methodol) 71(1):159–175
40. Sysoev O, Burdakov O, Grimvall A (2011) A segmentation-based algorithm for large-scale monotonic regression problems. J Comput Stat Data Anal 55:2463–2476
41. Sysoev O, Grimvall A, Burdakov O (2016) Bootstrap confidence intervals for large-scale multivariate monotonic regression problems. Commun Stat Simul Comput 45(3):1025–1040

42. Wang X, Berger JO (2016) Estimating shape constrained functions using Gaussian processes. SIAM/ASA J Uncertain Quantif 4(1):1–25
43. Wu J, Meyer M, Opsomer J (2015) Penalized isotonic regression. J Stat Plan Inference 161:12–24
44. Zhang J (2004) A simple and efficient monotone smoother using smoothing splines. J Nonparametr Stat 16(5):779–796

Oleg Sysoev is currently Associate Professor at the Division of Statistics and Machine Learning, Department of Computer and Information Science at Linköping University in Sweden. Oleg is also the head of the master's program in Statistics and Machine learning at Linkoping University. His research interests are monotonic regression and interpolation, data mining and machine learning methods (primarily decision trees, kernel smoothing, neural networks), data mining and machine learning in public health, machine learning for telecommunication systems, bootstrap methods, computational statistics, optimization.

Oleg Burdakov received his Ph.D. degree in 1980 from the Moscow University of Physics and Technology. He is a professor at Linköping University, Sweden, and an Affiliated Faculty of the Center for Applied Optimization, University of Florida, USA. He is the Editor-in-Chief of the Journal Optimization Methods and Software. His research interests include: numerical methods for solving optimization problems and systems of nonlinear equations, in particular, Newton-type, stable secant and interpolation methods, globalization strategies; cardinality constrained optimization; inverse problems; multilinear least-squares; nonsmooth optimization and equations; monotonic regression: data fitting and interpolation; hop-restricted shortest path and Steiner tree problems.