An algorithm for the estimation of a regression function by continuous piecewise linear functions

Adil Bagirov[1], Conny Clausen[2] and Michael Kohler[3]

[1] School of Information Technology and Mathematical Sciences, University of Ballarat, PO Box 663, Ballarat Victoria 3353 Australia, email: a.bagirov@ballarat.edu.au

[2] Department of Mathematics, Universität des Saarlandes, Postfach 151150, D-66041 Saarbrücken, Germany, email: clausen@math.uni-sb.de

[2] Department of Mathematics, Technische Universität Darmstadt, Schloßgartenstr. 7, D-64289 Darmstadt, Germany, email: kohler@mathematik.tu-darmstadt.de

Abstract

The problem of the estimation of a regression function by continuous piecewise linear functions is formulated as a nonconvex, nonsmooth optimization problem. Estimates are defined by minimization of the empirical L_2 risk over a class of functions, which are defined as maxima of minima of linear functions. An algorithm for finding continuous piecewise linear functions is presented. We observe that the objective function in the optimization problem is semismooth, quasidifferentiable and piecewise partially separable. The use of these properties allow us to design an efficient algorithm for approximation of subgradients of the objective function and to apply the discrete gradient method for its minimization. We present computational results with some simulated data and compare the new estimator with a number of existing ones.

Key words: nonsmooth optimization, nonparametric regression, subdifferential, semismooth functions.

1 Introduction

We consider the problem of estimating a multivariate regression function given a sample of the underlying distribution and develop an algorithm for the computation of continuous piecewise linear functions approximating such regression functions.

In applications usually no a priori information about the regression function is known, therefore it is necessary to apply nonparametric methods for this estimation problem. There are several established methods for nonparametric regression, including regression trees like CART [6], adaptive spline fitting like MARS [10] and least squares neural network estimates (Chapter 11 in [12]). All these methods minimize a kind of least squares risk of the regression estimate, either heuristically over

a fixed and very complex function space as for neural networks or over a stepwise defined data dependent space of piecewise constant functions or piecewise polynomials as for CART or MARS.

In this paper, we consider a rather complex function space consisting of continuous piecewise linear functions, over which we minimize an empirical least squares risk. Since each continuous piecewise linear function can be represented as a maxima of minima of linear functions [5, **?**], we use such a representation to estimate regression functions. We fit a linear spline function with *free* knots to the data. But in contrast to MARS, we do not need heuristics to choose these free knots, but use instead nonsmooth optimization methods to compute our estimate approximately in an application. Since continuous piecewise linear functions are, in general, nonsmooth and nonconvex, the resulting least squares risk is nonconvex and nonsmooth function. The Clarke subdifferential can be used to design algorithms for minimization of such functions [8]. However, the objective function is also non-regular and the Clarke sudifferential calculus cannot be used to estimate its subgradients. In this paper, we present an algorithm to approximate subgradients of the least squares risk function. Then we present the discrete gradient method which is based on those approximations to compute piecewise linear functions. We also present the computational results with simulated data and compare the proposed algorithm with various regression estimates.

The structure of the paper is as follows. Section 2 gives the definition of the estimate and the optimization reformulation of the problem of estimating a regression function. Properties of the objective function in the optimization problem are discussed in Section 3. An algorithm for approximation of subgradients is described in Section 4 and the minimization algorithm in Section 5. The implementation of the minimization algorithm is discussed in Section 6. We present numerical results with simulated data in Section 7. Section 8 concludes the paper.

2 Regression estimation

In *regression analysis* an $\mathbb{R}^p \times \mathbb{R}^1$-valued random vector (U, V) with $EV^2 < \infty$ is considered and the dependency of V on the value of U is of interest. More precisely, the goal is to find a function $\varphi : \mathbb{R}^p \to \mathbb{R}^1$ such that $\varphi(U)$ is a "good approximation" of V. In the sequel we assume that the main aim of the analysis is minimization of the mean squared prediction error or L_2 *risk*

$$\mathbf{E}\{|\varphi(U) - V|^2\}. \tag{1}$$

In this case the optimal function is the so-called *regression function* $m : \mathbb{R}^p \to \mathbb{R}^1$, $m(u) = \mathbf{E}\{V|U = u\}$. Indeed, let $\varphi : \mathbb{R}^p \to \mathbb{R}^1$ be an arbitrary

(measurable) function and denote the distribution of U by μ. Then

$$\begin{aligned}\mathbf{E}\{|\varphi(U)-V|^2\} &= \mathbf{E}\{((\varphi(U)-m(U))+(m(U)-V))^2\} \\ &= \mathbf{E}\{|\varphi(U)-m(U)|^2\}+\mathbf{E}\{|m(U)-V|^2\} \\ &= \mathbf{E}\{|m(U)-V|^2\}+\int|\varphi(u)-m(u)|^2\mu(du). \qquad (2)\end{aligned}$$

Here the second equation follows from

$$\begin{aligned}&\mathbf{E}\{(\varphi(U)-m(U))\cdot(m(U)-V)\} \\ &=\mathbf{E}\{(\varphi(U)-m(U))\cdot\mathbf{E}\{(m(U)-V)|U\}\}=0.\end{aligned}$$

Since the integral on the right-hand side of (2) is always nonnegative, (2) implies that the regression function is the optimal predictor in view of minimization of the L_2 risk:

$$\mathbf{E}\{|m(U)-V|^2\}=\min_{\varphi:\mathbb{R}^p\to\mathbb{R}^1}\mathbf{E}\{|\varphi(U)-V|^2\}. \qquad (3)$$

In addition, any function φ is a good predictor in the sense that its L_2 risk is close to the optimal value, if and only if the so-called L_2 *error*

$$\int|\varphi(u)-m(u)|^2\mu(du) \qquad (4)$$

is small. This motivates to measure the error caused by using a function φ instead of the regression function by the L_2 error (4).

In applications, usually the distribution of (U,V) (and hence also the regression function) is unknown. But often it is possible to observe a sample of the underlying distribution. This leads to the *regression estimation problem*. Here (U,V), (U_1,V_1), (U_2,V_2), ... are independent and identically distributed random vectors. The set of data

$$\mathcal{D}_l=\{(U_1,V_1),\ldots,(U_l,V_l)\}$$

is given, and the goal is to construct an estimate

$$m_l(\cdot)=m_l(\cdot,\mathcal{D}_l):\mathbb{R}^p\to\mathbb{R}^1$$

of the regression function such that the L_2 error

$$\int|m_l(u)-m(u)|^2\mu(du)$$

is small. For a detailed introduction to nonparametric regression we refer the reader to the monograph [11].

2.1 Least squares estimates

The regression function minimizes the L_2 risk (1) over the set of all measurable functions, so in principle it can be computed by solving a minimization problem. However, in an application the term to be minimized is unknown, because it depends on the unknown distribution of (U, V). For least squares estimates the given data is used to estimate the L_2 risk by the so-called empirical L_2 risk

$$\frac{1}{l}\sum_{i=1}^{l}|\varphi(U_i)-V_i|^2, \tag{5}$$

and the regression estimate is defined by minimizing (5). Minimization of (5) with respect to all measurable functions (as in (3)) leads to an estimate, which usually (at least if the values of $U_1, \ldots, U_l$ are distinct) interpolates the given data. Obviously, such an estimate is not a reasonable estimate for $m(u) = \mathbf{E}\{V|U = u\}$. In order to avoid this so-called overfitting, for least squares estimates, first a class $\mathcal{F}_l$ of functions $\varphi : \mathbb{R}^p \to \mathbb{R}^1$ is chosen and then the estimate is defined by minimizing the empirical L_2 risk over $\mathcal{F}_l$, i.e.,

$$m_l(\cdot) = \arg\min_{\varphi\in\mathcal{F}_l}\frac{1}{l}\sum_{i=1}^{l}|\varphi(U_i)-V_i|^2. \tag{6}$$

Here we assume that the minimum exists, however we do not require that it is unique.

2.2 Definition of the estimate

In the sequel we will use continuous piecewise linear functions to define $\mathcal{F}_l$. Since any continuous piecewise linear function can be represented as a max-min of finite number of linear functions (see [5]) we consider maxima of minima of linear functions. More precisely, let $K_l \in \mathbb{N}$ and $L_{1,l}, \ldots, L_{K_l,l} \in \mathbb{N}$ be parameters of the estimate and set

$$\mathcal{F}_l = \Bigg\{\varphi : \mathbb{R}^p \to \mathbb{R}^1 \,:\, \varphi(u) = \max_{k=1,\ldots,K_l}\min_{j=1,\ldots,L_{k,l}}\left(\langle x^{k,j}, u\rangle + y_{k,j}\right)(u \in \mathbb{R}^p),$$
$$\text{for some } x^{k,j} \in \mathbb{R}^p, y_{k,j} \in \mathbb{R}^1\Bigg\}$$

where

$$\langle x^{k,j}, u\rangle = \sum_{i=1}^{p} x_i^{k,j} u_i$$

denotes the scalar product between $x^{k,j} = (x_1^{k,j}, \ldots, x_p^{k,j})^T$ and $u = (u_1, \ldots, u_p)^T$. For this class of functions the estimate m_l is defined by (6).

2.3 Formulation of optimization problem

It follows from (5) that the estimation of a regression function by continuous piecewise linear function can be formulated as the following minimization problem:

$$\text{minimize} \quad F(x,y) = \frac{1}{l}\sum_{i=1}^{l}\left(\max_{k=1,\ldots,K_l}\min_{j=1,\ldots,L_{k,l}}\left(\langle x^{k,j}, U_i\rangle + y_{k,j}\right) - V_i\right)^2 \tag{7}$$

subject to

$$x^{k,j} \in \mathbb{R}^p,\ y_{k,j} \in \mathbb{R}^1,\ \ k = 1,\ldots,K_l,\ j = 1,\ldots,L_{k,l}.$$

Here

$$x = (x_1^{1,1},\ldots,x_p^{1,1},\ldots,x_1^{K_l,L_{K_l,l}},\ldots,x_p^{K_l,L_{K_l,l}}) \in \mathbb{R}^{q\times p},$$
$$y = (y_{1,1},\ldots,\ldots,y_{K_l,L_{K_l,l}}) \in \mathbb{R}^q$$

and

$$q = \sum_{k=1}^{K_l} L_{k,l}.$$

In the next section we study some properties of the function F.

3 Properties of F

In general, the objective function F in Problem (7) is nonsmooth and nonconvex. The number of its local minimizers is large, if the numbers l and q are large. The theory of Clarke generalized gradients can be applied to study such functions. Before describing properties of the function F we recall definitions of a Clarke subdifferential, a quasidiffential, semismooth and piecewise partially separable functions.

A function f, defined on $\mathbb{R}^n$, is called locally Lipschitz continuous if for any bounded subset $X \subset \mathbb{R}^n$ there exists an $R > 0$ such that

$$|f(x) - f(y)| \le R\|x - y\| \quad \forall x, y \in X.$$

In [7] (see, also [8]) Clarke introduced generalized gradients for Lipschitz functions. Since a locally Lipschitz function f is differentiable almost everywhere we can define for it a subdifferential by

$$\partial f(x) = \text{co}\left\{v \in \mathbb{R}^n : \exists (x^k \in D(f)) :\ x = \lim_{k\to\infty} x^k \text{ and } v = \lim_{k\to\infty}\nabla f(x^k)\right\},$$

here $D(f)$ denotes the set where f is differentiable, co denotes the convex hull of a set. The mapping $\partial f(x)$ is upper semicontinuous and bounded on bounded sets [8]. The generalized directional derivative of f at x in the direction g is defined as

$$f^0(x,g) = \limsup_{y\to x,\alpha\downarrow 0}\alpha^{-1}[f(y+\alpha g) - f(y)].$$

For the locally Lipschitz function f the generalized directional derivative exists and $f^0(x,g) = \max\{\langle v, g\rangle : v \in \partial f(x)\}$. f is called a Clarke regular function on $\mathbb{R}^n$, if it is directionally differentiable and $f'(x,g) = f^0(x,g)$ for all $x, g \in \mathbb{R}^n$, where $f'(x,g)$ is a derivative of the function f at the point x in the direction g:

$$f'(x,g) = \lim_{\alpha \downarrow 0} \alpha^{-1}[f(x+\alpha g) - f(x)].$$

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. For a point x to be a local minimizer of the function f on $\mathbb{R}^n$, it is necessary that $0 \in \partial f(x)$.

A function $f : \mathbb{R}^n \to \mathbb{R}^1$ is called semismooth at $x \in \mathbb{R}^n$, if it is locally Lipschitz at x and for each $g \in \mathbb{R}^n$ and for any sequences $\{t_k\} \subset \mathbb{R}^1,\ \{g^k\} \subset \mathbb{R}^n,\ \{v^k\} \subset \mathbb{R}^n$ such that $t_k \downarrow 0,\ g^k \to g,\ v^k \in \partial f(x + t_k g^k)$, the limit

$$\lim_{k \to \infty} \langle v^k, g\rangle$$

exists [13]. The semismooth function f is directionally differentiable and

$$f'(x,g) = \lim_{k \to \infty} \langle v^k, g\rangle, \quad v^k \in \partial f(x + t_k g^k).$$

A function f is called quasidifferentiable at a point x, if it is locally Lipschitz continuous, directionally differentiable at this point and there exist convex, compact sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ such that:

$$f'(x,g) = \max\left\{\langle u, g\rangle :\ u \in \underline{\partial} f(x)\right\} + \min\left\{\langle v, g\rangle :\ v \in \overline{\partial} f(x)\right\}.$$

The set $\underline{\partial} f(x)$ is called a subdifferential, the set $\overline{\partial} f(x)$ a superdifferential and the pair $[\underline{\partial} f(x), \overline{\partial} f(x)]$ a quasidifferential of the function f at a point x [9].

The function f is called a partially separable if there exists a family of $n \times n$ diagonal matrices $Q_i,\ i = 1, \ldots, M$ such that the function f can be represented as follows:

$$f(x) = \sum_{i=1}^{M} f_i(Q_i x).$$

We assume that the matrices Q_i are binary, that is they contain only 0 and 1 and the number of non-zero elements in the diagonal of the matrix Q_i is much smaller than n. In other terms, the function f is called partially separable if it can be represented as the sum of functions of a much smaller number of variables. If $M = n$ and $diag(Q_i) = e_i$ where e_i is the $i - th$ orth vector, then the function f is separable.

The function f is said to be piecewise partially separable if there exists a finite family of closed sets $D_1, \ldots, D_m$ such that $\bigcup_{i=1}^{m} D_i = \mathbb{R}^n$ and the function f is partially separable on each set $D_i,\ i = 1, \ldots, m$ (see [2]).

Now we can describe some of properties of the function F.

Proposition 1. *The function F is locally Lipschitz continuous.*

Proof: One can see from the definition of the function F that it is represented as a smooth composition of max-min type functions. More precisely, it is represented as a smooth composition of continuous nonconvex piecewise linear functions. Therefore it is nonconvex. Since both smooth and continuous piecewise linear functions are locally Lipschitz continuous the function F is also locally Lipschitz continuous. $\square$

Remark 1. In general, nonconvex piecewise linear functions are not regular, then the function F, in general, is not regular. It follows from Proposition 1 that F is subdifferentiable, however it is nonregular. The Clarke subdifferential calculus for such functions exists in the form of inclusions and such a calculus cannot be used to estimate subgradients of the function F. Therefore computations of even one subgradient of this function is a quite difficult task. In Section 4 we describe an algorithm for approximation of subgradients of the function F.

Proposition 2. *The function F is quasidifferentiable and its subdifferential and superdifferential are polytopes.*

Proof: Consider the function

$$\psi_{i,k}(x,y) = \min_{j=1,\ldots,L_{k,l}} \left\{ \langle x^{k,j}, U_i \rangle + y_{k,j} \right\}, \quad i = 1,\ldots,l, \ k = 1,\ldots,K_l.$$

This function can be represented as the difference of two convex functions $\psi^1_{i,k}$ and $\psi^2_{i,k}$ as follows:

$$\psi_{i,k}(x,y) = \psi^1_{i,k}(x,y) - \psi^2_{i,k}(x,y),$$

where

$$\psi^1_{i,k}(x,y) = \sum_{j=1}^{L_{k,l}} \left(\langle x^{k,j}, U_i \rangle + y_{k,j} \right),$$

$$\psi^2_{i,k}(x,y) = \max_{j=1,\ldots,L_{k,l}} \sum_{t=1,t\neq j}^{L_{k,l}} \left(\langle x^{k,t}, U_i \rangle + y_{k,t} \right).$$

The function $\psi^1_{i,k}$ is linear and the function $\psi^2_{i,k}$ is piecewise linear convex. The subdifferentials of the function $\psi^2_{i,k}$ at any point (x,y), $x \in \mathbb{R}^{q\times p}$, $y \in \mathbb{R}^q$ are polytopes, that is they can be expressed as a convex combination of finite number of points.

Now consider the following function

$$\varphi_i(x,y) = \max_{k=1,\ldots,K_l} \psi_{i,k}(x,y).$$

This function can be represented as the difference of two convex functions φ_i^1 and φ_i^2 as follows:

$$\varphi_i(x,y) = \varphi_i^1(x,y) - \varphi_i^2(x,y)$$

where

$$\varphi_i^1(x,y) = \max_{k=1,\ldots,K_l} \left(\psi_{i,k}^1(x,y) + \sum_{j=1, j\neq k}^{K_l} \psi_{i,j}^2(x,y) \right),$$

$$\varphi_i^2(x,y) = \sum_{j=1}^{K_l} \psi_{i,k}^2(x,y).$$

Both functions φ_i^1 and φ_i^2 are convex piecewise linear and their subdiffierentials are polytopes at any point $(x,y), x \in \mathbb{R}^{q\times p},\ y \in \mathbb{R}^q$. This means that the function φ_i is quasidifferentiable and its subdifferentials and superdifferentials are polytopes at any point.

Finally, the function F is smooth composition of the functions φ_i and therefore it is quasidifferentiable [9]. Since for any smooth functions subdifferential is the singleton set at any point, the function F is quasidifferentiable and its subdifferential and superdifferential are polytopes. □

Proposition 3. *The function F is semismooth.*

Proof: The proof follows from the facts that linear functions are semismooth, minimum of semismooth functions is again semismooth, maximum of semismooth functions is also semismooth and finally, the smooth composition of semismooth functions is semismooth [13]. □

Proposition 4. *The function F is piecewise partially separable.*

Proof: Linear functions are separable. Maximum and minimum of linear functions is piecewise separable and finally, smooth composition of continuous piecewise linear functions is piecewise partially separable (see [2]). Therefore, the function F is piecewise partially separable. More precisely, for any $k \in \{1,\ldots,K_l\}$ and $j \in \{1,\ldots,L_{k,l}\}$ there exists a set $D_{kj} \subset \mathbb{R}^n$ such that the function F in this set can be represented as follows:

$$F(x,y) = \frac{1}{l} \sum_{i=1}^{l} \left(\langle x^{k,j}, U_i \rangle + y_{k,j} - V_i \right)^2.$$

It is clear that

$$\bigcup_{k=1}^{K_l} \bigcup_{j=1}^{L_{k,l}} D_{kj} = \mathbb{R}^n,$$

however some of sets D_{kj} can be empty. □

4 Approximation of subgradients

In this section a technique to approximate subgradients of the function F is described. This approach is introduced in [3, 4]. All necessary proofs also can be found in these papers.

We consider a function f defined on $\mathbb{R}^n$ and assume that this function is quasidifferentiable. We also assume that both sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ are polytopes at any $x \in \mathbb{R}^n$. We denote by Φ the class of all semismooth, quasidifferentiable functions defined on $\mathbb{R}^n$, whose subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$. Results from the previous section show that the function F belongs to this class.

Let $G = \{e \in \mathbb{R}^n : e = (e_1, \ldots, e_n), |e_j| = 1,\ j = 1, \ldots, n\}$ be a set of all vertices of the unit hypercube in $\mathbb{R}^n$. We take $e \in G$ and consider the sequence of n vectors $e^j = e^j(\alpha),\ j = 1, \ldots, n$ with $\alpha \in (0, 1]$:

$$\begin{array}{rcl} e^1 & = & (\alpha e_1, 0, \ldots, 0), \\ e^2 & = & (\alpha e_1, \alpha^2 e_2, 0, \ldots, 0), \\ \ldots & = & \ldots\ldots\ldots \\ e^n & = & (\alpha e_1, \alpha^2 e_2, \ldots, \alpha^n e_n). \end{array}$$

Let $e \in G$ be a given vector and $\lambda > 0,\ \alpha \in (0, 1]$ be given numbers. Consider the following points

$$x^0 = x, \quad x^j = x^0 + \lambda e^j(\alpha), \quad j = 1, \ldots, n.$$

It is clear that

$$x^j = x^{j-1} + (0, \ldots, 0, \lambda \alpha^j e_j, 0, \ldots, 0), \quad j = 1, \ldots, n.$$

Let $v = v(\alpha, \lambda) \in \mathbb{R}^n$ be a vector with the following coordinates:

$$v_j = (\lambda \alpha^j e_j)^{-1} \left[f(x^j) - f(x^{j-1}) \right], \quad j = 1, \ldots, n. \tag{8}$$

For any fixed $e \in G$ and $\alpha > 0$ we introduce the set:

$$V(e, \alpha) = \left\{ w \in \mathbb{R}^n : \ \exists (\lambda_k \to +0, \ \ k \to +\infty), \ \ w = \lim_{k \to +\infty} v(\alpha, \lambda_k) \right\}.$$

Proposition 5. [3, 4]. *Assume that $f \in \Phi$. Then there exists $\alpha_0 \in (0, 1]$ such that*

$$V(e, \alpha) \subset \partial f(x),\ \forall\ \alpha \in (0, \alpha_0].$$

Remark 2. It follows from Proposition 5 that in order to approximate subgradients of the function F one can choose a vector $e \in G$, sufficiently small $\alpha > 0,\ \lambda > 0$ and apply (8) to compute a vector $v(\alpha, \lambda)$. This vector is an approximation to a subgradient.

4.1 Computation of subdifferentials

In this subsection we consider an algorithm for the computation of subdifferentials of the function F. This algorithm is based on the notion of a discrete gradient. We start with the definition of the discrete gradient, which was introduced in [1].

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\}, \quad P = \{z : \quad z : \mathbb{R}^+ \to \mathbb{R}^+, \ \beta^{-1} z(\beta) \downarrow 0, \ \beta \downarrow 0\}.$$

Here S_1 is the unit sphere and P is the set of univariate positive infinitesimal functions. We take any $g \in S_1, \ e \in G$ and a positive number $\alpha \in (0, 1]$. Then we define $|g_i| = \max\{|g_k|, \quad k = 1, \ldots, n\}$ and the sequence of n vectors $e^j(\alpha), \ j = 1, \ldots, n$. For given $x \in \mathbb{R}^n$ and $z \in P$ consider a sequence of $n + 1$ points:

$$\begin{array}{lll} x^0 = & x+ & \lambda g, \\ x^1 = & x^0+ & z(\lambda) e^1(\alpha), \\ \ldots = & \ldots & \ldots \\ x^n = & x^0+ & z(\lambda) e^n(\alpha). \end{array}$$

Definition 1. *[1] The discrete gradient of the function f at the point $x \in \mathbb{R}^n$ is the vector $\Gamma(x, g, e, z, \lambda, \alpha) = (\Gamma_1, \ldots, \Gamma_n) \in \mathbb{R}^n, g \in S_1$ with the following coordinates:*

$$\Gamma_j = [z(\lambda)\alpha^j e_j)]^{-1} \left[f(x^j) - f(x^{j-1}) \right], \quad j = 1, \ldots, n, \quad j \neq i,$$

$$\Gamma_i = (\lambda g_i)^{-1} \left[f(x + \lambda g) - f(x) - \lambda \sum_{j=1, j \neq i}^{n} \Gamma_j g_j \right].$$

It follows from Definition 1 that

$$f(x + \lambda g) - f(x) = \lambda \langle \Gamma(x, g, e, z, \lambda, \alpha), g \rangle \tag{9}$$

for all $g \in S_1, \ e \in G, \ z \in P, \ \lambda > 0, \ \alpha > 0$.

Remark 3. One can see that the discrete gradient is defined with respect to a given direction $g \in S_1$ and in order to compute it, first we define a sequence of points $x^0, \ldots, x^n$ and compute the values of the function f at these points that is we compute $n + 2$ values of this function including the point x. $n - 1$ coordinates of the discrete gradient are defined similar to those of the vector $v(\alpha, \lambda)$ and i-th coordinate is defined so that to satisfy the equality (9) which can be considered as some version of the mean value theorem.

Remark 4. Since the function F is piecewise partially separable we will use a special scheme described in [2] to compute its discrete gradients. This scheme allows us to use only two evaluations instead of $n + 2$ evaluations of the function F to compute one discrete gradient.

For a given $\alpha > 0$ we define the following set:

$$B(x,\alpha) = \{v \in \mathbb{R}^n : \ \exists (g \in S_1,\ e \in G,\ z_k \in P,\ \lambda_k \in \mathbb{R}^1) :\ z_k \downarrow 0,\ \lambda_k \downarrow 0,\ k \to +\infty$$
$$\text{and } v = \lim_{k \to +\infty} \Gamma(x, g, e, z_k, \lambda_k, \alpha)\}. \tag{10}$$

Proposition 6. *[3, 4] Assume that $f \in \Phi$. Then there exists $\alpha_0 \in (0,1]$ such that*

$$co\, B(x,\alpha) \subset \partial f(x),\ \forall\ \alpha \in (0, \alpha_0].$$

Remark 5. Proposition 6 shows how one can approximate the subdifferential of the function F. However, in general, the computation of the set $B(x,\alpha)$ or its approximation is not easy. In the next section we describe an algorithm, where only a few discrete gradients are computed to find descent directions of the function F.

5 The discrete gradient method

In this section we describe the discrete gradient method for minimizing the function F. An important step in this method is the computation of descent directions. Therefore we start with the description of an algorithm for finding descent directions.

5.1 Computation of descent directions

Let $z \in P, \lambda > 0, \alpha \in (0,1]$, the number $c \in (0,1)$ and a tolerance $\delta > 0$ be given.

Algorithm 1. Computation of the descent direction.

Step 1. Choose any $g^1 \in S_1, e \in G$, compute $i = \text{argmax}\,\{|g_j|,\ j = 1, \dots, n\}$ and a discrete gradient $v^1 = \Gamma(x, g^1, e, z, \lambda, \alpha)$. Set $\overline{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Compute the vector $\|w^k\|^2 = \min\{\|w\|^2 : w \in \overline{D}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{11}$$

then stop. Otherwise go to Step 3.

Step 3. Compute the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4. If

$$f(x + \lambda g^{k+1}) - f(x) \leq -c\lambda \|w^k\|, \tag{12}$$

then stop. Otherwise go to Step 5.

Step 5. Compute $i = \text{argmax}\,\{|g_j^{k+1}| :\ j = 1, \dots, n\}$ and a discrete gradient

$$v^{k+1} = \Gamma(x, g^{k+1}, e, z, \lambda, \alpha),$$

construct the set $\overline{D}_{k+1}(x) = \text{co}\,\{\overline{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k + 1$ and go to Step 2.

Some explanations to Algorithm 1 are necessary. In Step 1 we compute the discrete gradient in an initial direction $g^1 \in \mathbb{R}^n$. The distance between the convex hull $\overline{D}_k(x)$ of all computed discrete gradients and the origin is computed in Step 2. This problem can be solved using the algorithm from [15]. If this distance is less than the tolerance $\delta > 0$, then we accept the point x as an approximate stationary point (Step 2), otherwise we compute another search direction in Step 3. In Step 4 we check whether this direction is a descent direction. If it is, we stop and the descent direction has been computed, otherwise we compute another discrete gradient in this direction in Step 5 and update the set $\overline{D}_k(x)$. At each iteration k we improve the approximation of the subdifferential of the function f.

Algorithm 1 terminates after a finite number of iterations [4].

5.2 The method

Let sequences $\delta_k > 0,\ z_k \in P,\ \lambda_k > 0,\ \delta_k \downarrow 0,\ z_k \downarrow 0,\ \lambda_k \downarrow 0,\ k \to +\infty$, sufficiently small number $\alpha > 0$ and numbers $c_1 \in (0,1), c_2 \in (0, c_1]$ be given.

Algorithm 2. The discrete gradient method

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x_s^k = x^k$.

Step 3. Apply Algorithm 1 for the computation of the descent direction at $x = x_s^k, \delta = \delta_k, z = z_k, \lambda = \lambda_k, c = c_1$. This algorithm terminates after a finite number of iterations $r > 0$. As a result we get the set $\overline{D}_r(x_s^k)$ and an element v_s^k such that

$$\|v_s^k\|^2 = \min\{\|v\|^2 :\ v \in \overline{D}_r(x_s^k)\}.$$

Furthermore either $\|v_s^k\| \le \delta_k$ or for the search direction $g_s^k = -\|v_s^k\|^{-1} v_s^k$

$$f(x_s^k + \lambda_k g_s^k) - f(x_s^k) \le -c_1 \lambda_k \|v_s^k\|. \tag{13}$$

Step 4. If

$$\|v_s^k\| \le \delta_k \tag{14}$$

then set $x^{k+1} = x_s^k, k = k + 1$ and go to Step 2. Otherwise go to Step 5.

Step 5. Construct the following iteration $x_{s+1}^k = x_s^k + \sigma_s g_s^k$, where σ_s is defined as follows

$$\sigma_s = \text{argmax}\left\{\sigma \ge 0 :\ f(x_s^k + \sigma g_s^k) - f(x_s^k) \le -c_2 \sigma \|v_s^k\|\right\}.$$

Step 6. Set $s = s + 1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \{x \in \mathbb{R}^n : f(x) \le f(x^0)\}$.

Proposition 7. [4] *Assume that the function $f \in \Phi$ and the set $M(x^0)$ is bounded for any $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Remark 6. It follows from Proposition 7 that the discrete gradient method can be applied to minimize the function F.

6 Implementation

In this section we describe conditions for the implementation of Algorithm 2 to solve Problem (7). The following conditions have been chosen:

1. Since Algorithm 1 can compute descent directions for any values of $\lambda > 0$ we take $\lambda_0 > 1$, some $\beta \in (0, 1)$ and update $\lambda_k,\ k \geq 1$ by the formula $\lambda_k = \beta^k \lambda_0,\ \ k \geq 1$. In our computations $\lambda_0 = 3$ and $\beta = 0.6$.

2. It follows from (13) and the condition $c_2 \leq c_1$ that always $\sigma_s \geq \lambda_k$ and therefore $\lambda_k > 0$ is a lower bound for σ_s. This leads to the following rule for the computation of σ_s. We define a sequence $\theta_m = m\lambda_k,\ \ m \geq 1$ and σ_s is defined as the largest θ_m satisfying the inequality in Step 5. In our computations $c_1 \in (0.2, 0.5)$ and $c_2 = 0.001$.

3. In our computations $\alpha = 1,\ \delta_k = 10^{-7}$ and $z_k(\lambda) = \lambda^2$ for all k.

4. In our applications we choose the number of linear functions considered in the maxima and the minima in a data-dependent way by splitting of the sample. We split the sample of size l in a learning sample of size $l_{learning} < l$ and a testing sample of size $l_{testing} = l - l_{learning}$. We use the learning sample to define for a fixed number of minima functions K_l and a fixed number of linear functions under minima $L_{K_l,l}$ an estimate $m_{l_{learning},K,L}$, and compute the empirical L_2 risk of this estimate on the testing sample. Since the testing sample is independent of the learning sample, this gives us an unbiased estimate of the L_2 risk of $m_{l_{learning},K,L}$. Then we choose K and L by minimizing this estimate with respect to K_l and $L_{K_l,l}$. In the sequel we use $l \in \{500, 5000\},\ l_{testing} = l_{learning} = l/2$ and $K_l \in \{1, \ldots, 5\},\ L_{K_l,l} \in \{1, \ldots, 5\}$.

7 Application to simulated data

In order to compare the estimates proposed in this paper with other nonparametric regression estimates we made a simulation study. Here we define (U, V) by

$$V = m(U) + \sigma \cdot \epsilon,$$

where ϵ is standard normally distributed and independent of U and $\sigma \geq 0$, and where U is uniformly distributed on $[-2,2]^p$. For the noise level σ we use three different values: 0, 0.5 and 1. We generate data sets of two different sample sizes, namely $l = 500$ and $l = 5000$.

For $p = 1$ we compare our estimate with kernel estimates (with Gaussian kernel) (see Chapter 5, [11]), local linear kernel estimates (see Section 5.4, [11]), smoothing splines (see Chapter 20, [11]), neural networks and regression trees (as implemented in the freely available statistics software R [14]) by applying every one of these six estimates to samples of the above distributions. Since for $p > 1$ not all of these estimates are easily applicable in R, we compare for $p > 1$ our estimate only with neural networks and regression trees (again by applying every one of these three estimates to samples of the above distributions). In all cases we choose the smoothing parameter of the estimates by splitting of the sample, where for each simulation the size of the training sample and the testing sample is $l/2$.

In order to compute the L_2 errors of the estimates, we use Monte Carlo integration, i.e., we approximate

$$\int |m_l(u) - m(u)|^2 \mu(du) = \mathbf{E}\{|m_l(U) - m(U)|^2 | \mathcal{D}_l\}$$

by

$$\frac{1}{N} \sum_{j=1}^{N} |m_l(\tilde{U}_j) - m(\tilde{U}_j)|^2,$$

where the random variables $\tilde{U}_1, \tilde{U}_2, \ldots$ are i.i.d. with distribution $\mu = \mathbf{P}_U$ and independent of $\mathcal{D}_l$. In the sequel we use $N = 3000$. Since this error is a random variable itself, we repeat the experiment 25 times with independent realizations of the sample, and report the mean and the standard deviation of the Monte Carlo estimates of the L_2 error.

First we consider the case $p = 1$ and consider the following four different regression functions:

- $m_1(u) = 2 * \max(1, \min(3 + 2 * u, 3 - 8 * u))$,
- $m_2(u) = \begin{cases} 1 & , u \leq 0, \\ 3 & , \text{ else}, \end{cases}$
- $m_3(u) = \begin{cases} 10 * \sqrt{-u} * \sin(8 * \pi * u) & , -0.25 \leq u < 0, \\ 0 & , \text{ else}, \end{cases}$
- $m_4(u) = 3 * \sin(\pi * u/2)$,

Figure 1 shows the four different univariate regression functions, Figure 2 shows

Figure 1: The four univariate regression functions.

Figure 2: The four univariate regression functions (solid lines) together with the max-min-estimate (dash lines) applied to a sample with variance $\sigma = 0.2$ and sample size $l = 500$.

these function together with our max-min-estimate applied to a sample with variance $\sigma = 0.2$ and sample size $l = 500$. In Tables 1 to 4 we report the error values for our maxmin estimate and the other five univariate regression estimates which are applied to the simulated data described above. From these tables we can see that in case of the distributions considered above our estimate outperforms the other estimates if the sample size is large and the regression function is not globally smooth like the fourth regression function.

Next we consider the case $p = 2$ and the following three regression functions:

- $m_5(u_1, u_2) = u_1 * \sin(u_1^2) - u_2 * \sin(u_2^2)$,
- $m_6(u_1, u_2) = \frac{4}{1+4*u_1^2+4*u_2^2}$,
- $m_7(u_1, u_2) = 6 - 2 * \min(3, 4 * u_1^2 + 4 * |u_2|)$,

Figures 3-5 show the three bivariate regression functions together with our max-min-estimate applied to a sample with variance $\sigma = 0.2$ and sample size $l = 5000$.

Figure 3: The bivariate regression function m_5 together with our max-min-estimate applied to a sample with variance $\sigma = 0.2$ and sample size $l = 5000$.

In Table 5 we compare our maxmin estimate with regression trees and neural networks. As above we report the error values for our maxmin estimate and the other two bivariate regression estimates which are applied to the simulated data described above. Here our estimate is most of the time better than regression trees and sometimes better and sometimes worse than neural networks.

Finally, we consider the case $p = 10$ where we use the following four regression functions for our simulations:

l	σ	est. 1	est. 2	est. 3	est. 4	est. 5	maxmin est.
500	0	0.0022	0.0005	0.0001	0.0020	0.0347	**0.0000**
		(0.0017)	(0.0004)	(0.0001)	(0.0004)	(0.0062)	(0.0000)
	0.5	0.0288	0.0278	0.0242	0.0161	0.0798	**0.0093**
		(0.0075)	(0.0078)	(0.0065)	(0.0039)	(0.0123)	(0.0048)
	1	0.0741	0.0816	0.0760	0.0438	0.2204	**0.0408**
		(0.0268)	(0.0389)	(0.0327)	(0.0206)	(0.0445)	(0.0254)
5000	0	0.0003	0.0003	**0.0000**	0.0006	0.0009	**0.0000**
		(0.0000)	(0.0000)	(0.0000)	(0.0002)	(0.0001)	(0.0000)
	0.5	0.0044	0.0043	0.0038	0.0030	0.0105	**0.0007**
		(0.0011)	(0.0009)	(0.0007)	(0.0008)	(0.0017)	(0.0005)
	1	0.0131	0.0121	0.0118	0.0091	0.1358	**0.0028**
		(0.0032)	(0.0036)	(0.0030)	(0.0020)	(0.0232)	(0.0015)

Table 1. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of kernel estimates (est. 1), local linear kernel estimates (est. 2), smoothing splines (est. 3), neural networks (est. 4) and regression trees (est. 4). The regression function is m_1.

- $m_8(u_1, ..., u_{10}) = \sum_{j=1}^{10} (-1)^{j-1} * u_j * \sin(u_j^2)$,
- $m_9(u_1, ..., u_{10}) = m_7(u_1, u_2)$,
- $m_{10}(u_1, ..., u_{10}) = m_6(u_1 + ... + u_5, u_6 + ... + u_{10})$,
- $m_{11}(u_1, ..., u_{10}) = m_2(u_1 + ... + u_{10})$.

Again we compare our maxmin estimate with regression trees and neural networks. In Table 6 we report the error values for our maxmin estimate and the other two multivariate regression estimates which are applied to the simulated data described above. Here none of the estimate is able to estimate m_8 well, the other method outperform our estimate for m_9 (which is a very simple function depending in fact only of two of the components of the predictor variable), but our estimate clearly outperforms the other estimates in case of $l = 5000$ and m_{10} and for $l \in \{500, 5000\}$ in case of m_{11}.

l	σ	est. 1	est. 2	est. 3	est. 4	est. 5	maxmin est.
500	0	0.0078	0.0096	0.0072	0.0110	0.0087	**0.0045**
		(0.0486)	(0.0047)	(0.0051)	(0.0047)	(0.0108)	(0.0046)
	0.5	0.0365	0.0396	0.0375	0.0165	0.0608	**0.0156**
		(0.0100)	(0.0087)	(0.0083)	(0.0052)	(0.0153)	(0.0110)
	1	0.0684	0.0806	0.0746	**0.0288**	0.2260	0.0431
		(0.0160)	(0.0171)	(0.0170)	(0.0184)	(0.0489)	(0.0240)
5000	0	0.0058	0.0074	0.0026	0.0040	0.0009	**0.0007**
		(0.0011)	(0.0013)	(0.0007)	(0.0009)	(0.0018)	(0.0011)
	0.5	0.0106	0.0119	0.0110	0.0051	0.0033	**0.0013**
		(0.0013)	(0.0013)	(0.0011)	(0.0009)	(0.0032)	(0.0008)
	1	0.0219	0.0241	0.0226	0.0076	0.1539	**0.0041**
		(0.0039)	(0.0039)	(0.0039)	(0.0021)	(0.0203)	(0.0022)

Table 2. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of kernel estimates (est. 1), local linear kernel estimates (est. 2), smoothing splines (est. 3), neural networks (est. 4) and regression trees (est. 4). The regression function is m_2.

8 Conclusion

In this paper we proposed an algorithm to compute continuous piecewise linear estimation of a regression function. This problem was formulated as an optimization problem where the objective function is nonconvex and nonsmooth. Moreover, it is nonregular and the Clarke calculus cannot be applied to compute subgradients of such functions. We proposed the special scheme to approximate the subgradients. The discrete gradient method based on those approximate subgradients is suggested to minimize the objective function. We present the results with simulated data and compare this approach with the number of other estimates. Numerical results confirm that the max-min estimate is effective for the estimation of regression functions.

Acknowledgement

Dr. Adil Bagirov is the recipient of an Australian Research Council Australian Research Fellowship (Project number: DP 0666061).

l	σ	est. 1	est. 2	est. 3	est. 4	est. 5	maxmin est.
500	0	0.0539	0.0450	0.0052	**0.0081**	0.1241	0.0234
		(0.0502)	(0.0402)	(0.0041)	(0.0064)	(0.0610)	(0.0585)
	0.5	0.0879	0.0922	0.0748	**0.0214**	0.1761	0.0255
		(0.0238)	(0.0383)	(0.0183)	(0.0101)	(0.0477)	(0.0178)
	1	0.2450	0.2749	0.2426	**0.0814**	0.3506	0.1201
		(0.0644)	(0.0735)	(0.0645)	(0.0490)	(0.0657)	(0.0556)
5000	0	0.0151	0.0175	0.0002	0.0010	0.0066	**0.0006**
		(0.0025)	(0.0054)	(0.0002)	(0.0003)	(0.0019)	(0.0001)
	0.5	0.0202	0.0220	0.0095	0.0030	0.0344	**0.0022**
		(0.0036)	(0.0060)	(0.0014)	(0.0008)	(0.0070)	(0.0006)
	1	0.0351	0.0357	0.0286	0.0080	0.1875	**0.0068**
		(0.0044)	(0.0047)	(0.0040)	(0.0041)	(0.0173)	(0.0021)

Table 3. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of kernel estimates (est. 1), local linear kernel estimates (est. 2), smoothing splines (est. 3), neural networks (est. 4) and regression trees (est. 4). The regression function is m_3.

References

[1] A.M. Bagirov, "Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices," In: A. Eberhard et al. (eds.) Progress in Optimization: Contribution from Australia, Kluwer Academic Publishers: Dordrecht, pp. 147-175, 1999.

[2] A.M. Bagirov and J. Ugon, J. "Piecewise partially separable functions and a derivative-free method for large-scale nonsmooth optimization," Journal of Global Optimization, vol. 35, pp. 163-195, 2006.

[3] A.M. Bagirov, M. Ghosh and D. Webb, "A derivative-free method for linearly constrained nonsmooth optimization," Journal of Industrial and Management Optimization, vol. 2(3): pp. 319-338, 2006.

[4] A.M. Bagirov, B. Karasozen and M. Sezer, "Discrete gradient method: a derivative free method for nonsmooth optimization," Journal of Optimization Theory and Applications, accepted for publication.

l	σ	est. 1	est. 2	est. 3	est. 4	est. 5	maxmin est.
500	0	0.0010	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0041
		(0.0003)	(0.0000)	(0.0000)	(0.0000)	(0.0492)	(0.0125)
	0.5	0.0188	0.0084	**0.0072**	0.0129	0.0813	0.0207
		(0.0058)	(0.0027)	(0.0034)	(0.0060)	(0.0113)	(0.0069)
	1	0.0622	**0.0316**	0.0318	0.0564	0.2157	0.0634
		(0.0260)	(0.0157)	(0.0161)	(0.0321)	(0.0404)	(0.0192)
5000	0	0.0001	**0.0000**	**0.0000**	**0.0000**	0.0005	0.0037
		(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0000)	(0.0017)
	0.5	0.0031	0.0014	**0.0010**	0.0014	0.0190	0.0061
		(0.0006)	(0.0004)	(0.0004)	(0.0005)	(0.0017)	(0.0014)
	1	0.0085	0.0040	**0.0034**	0.0049	0.0533	0.0113
		(0.0022)	(0.0018)	(0.0014)	(0.0023)	(0.0082)	(0.0035)

Table 4. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of kernel estimates (est. 1), local linear kernel estimates (est. 2), smoothing splines (est. 3), neural networks (est. 4) and regression trees (est. 4). The regression function is m_4.

[5] S.G. Bartels, L. Kuntz and S. Sholtes, "Continuous selections of linear functions and nonsmooth critical point theory," Nonlinear Analysis, TMA, vol. 24, pp. 385-407, 1995.

[6] L. Breiman, J.H. Friedman, R.H. Olshen and C.J. Stone, Classification and regression trees, Wadsworth: Belmont, CA, 1984.

[7] F.H. Clarke, "Generalized gradients and applications," Trans. Amer. Math. Soc., vol. 205, pp. 247 - 262, 1975.

[8] F.H. Clarke, Optimization and Nonsmooth Analysis, New York: John Wiley, 1983.

[9] V.F. Demyanov and A.M. Rubinov, Constructive Nonsmooth Analysis, Peter Lang: Frankfurt am Main, 1995.

[10] J.H. Friedman, "Multivariate adaptive regression splines (with discussion)," Annals of Statistics, vol. 19, pp. 1-141, 1991.

Figure 4: The bivariate regression function m_6 together with our max-min-estimate applied to a sample with variance $\sigma = 0.2$ and sample size $l = 5000$.

Figure 5: The bivariate regression function m_7 together with our max-min-estimate applied to a sample with variance $\sigma = 0.2$ and sample size $l = 5000$.

[11] V.V. Gorokhovik, O.I. Zorko and G. Birkhoff, "Piecewise affine functions and polyhedral sets", Optimization, vol. 31(3), pp. 209-221, 1994.

[12] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk, H., A Distribution-Free Theory of Nonparametric Regression, Springer Series in Statistics, Springer: Heldelberg, 2002.

[13] T. Hastie, R. Tibshirani and J. Friedman, The elements of statistical learning, Springer-Verlag: New York, 2001.

[14] R. Mifflin, "Semismooth and semiconvex functions in constrained optimization," SIAM Journal on Control and Optimization, vol. 15, pp. 957-972, 1977.

[15] The R Project for Statistical Computing, available on: www.r-project.org.

[16] P.H. Wolfe, "Finding the nearest point in a polytope," Mathematical Programming, vol. 11, pp. 128-149, 1976.

	l	500			5000		
	σ	0	0.5	1	0	0.5	1
m_5	est. 1	**0.0001** (0.0000)	**0.0657** (0.0206)	0.2897 (0.1105)	**0.0000** (0.0000)	**0.0049** (0.0021)	**0.02284** (0.0103)
	est. 2	0.3718 (0.0551)	0.4128 (0.0458)	0.5610 (0.0922)	0.0613 (0.0070)	0.1002 (0.0088)	0.1872 (0.0169)
	maxmin est.	0.0796 (0.0170)	0.1449 (0.0310)	**0.2280** (0.0490)	0.0593 (0.0090)	0.0700 (0.0064)	0.0889 (0.0104)
m_6	est. 1	**0.0015** (0.0006)	0.0822 (0.0211)	0.2026 (0.0438)	**0.0001** (0.0000)	**0.0110** (0.0034)	0.0339 (0.0076)
	est. 2	0.0817 (0.0202)	**0.0123** (0.0261)	0.2062 (0.0621)	0.0083 (0.0006)	0.0312 (0.0041)	0.0607 (0.0073)
	maxmin est.	0.0134 (0.0040)	0.0540 (0.0135)	**0.1543** (0.0629)	0.0066 (0.0018)	0.0137 (0.0015)	**0.0293** (0.0048)
m_7	est. 1	**0.0298** (0.0108)	0.1874 (0.0617)	0.4884 (0.1198)	**0.0078** (0.0011)	0.0253 (0.0033)	0.0699 (0.0112)
	est. 2	0.3034 (0.1547)	0.3175 (0.1967)	0.3757 (0.1820)	0.0484 (0.0071)	0.0610 (0.0081)	0.0902 (0.0166)
	maxmin est.	0.0325 (0.0087)	**0.0868** (0.0321)	**0.1734** (0.0660)	0.0136 (0.0036)	**0.0176** (0.0046)	**0.0260** (0.0055)

Table 5. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of neural networks (est. 1) and regression trees (est. 2). The regression function is m_5, m_6 or m_7, respectively.

	l	500			5000		
	σ	0	0.5	1	0	0.5	1
m_8	est. 1	5.5527 (0.1840)	5.4825 (0.2261)	5.6506 (0.2479)	4.7018 (0.1304)	4.6583 (0.1361)	4.7093 (0.1071)
	est. 2	5.6535 (0.1817)	5.6297 (0.2013)	5.7852 (0.2513)	5.0029 (0.1515)	4.9726 (0.1431)	5.0189 (0.1139)
	maxmin est.	**4.4715** (0.1884)	**4.4842** (0.1593)	**4.5392** (0.1532)	**3.7220** (0.1526)	**3.7106** (0.1403)	**3.7852** (0.1250)
m_9	est. 1	**0.0265** (0.0081)	**0.1790** (0.0531)	0.4805 (0.0917)	**0.0079** (0.0014)	**0.0247** (0.0023)	**0.0680** (0.0097)
	est. 2	0.3011 (0.1826)	0.2980 (0.1073)	**0.3756** (0.2008)	0.0477 (0.0071)	0.0587 (0.0078)	0.0901 (0.0131)
	maxmin est.	0.6216 (0.1049)	0.8003 (0.1255)	0.9121 (0.0928)	0.0279 (0.0133)	0.0521 (0.0085)	0.1471 (0.0358)
m_{10}	est. 1	0.2064 (0.0231)	0.2122 (0.0147)	0.2284 (0.0284)	0.2018 (0.0116)	0.1982 (0.0185)	0.2061 (0.0190)
	est. 2	0.2033 (0.0226)	**0.2024** (0.0134)	**0.2053** (0.0263)	0.2028 (0.0116)	0.1987 (0.0186)	0.2039 (0.0190)
	maxmin est.	**0.1893** (0.0215)	0.2577 (0.0697)	0.2944 (0.0757)	**0.0236** (0.0035)	**0.0502** (0.0066)	**0.1135** (0.0232)
m_{11}	est. 1	0.8902 (0.0180)	0.9057 (0.0286)	0.9270 (0.0381)	0.8711 (0.0126)	0.8766 (0.0126)	0.8738 (0.0139)
	est. 2	0.9659 (0.0244)	0.9745 (0.0281)	1.0037 (0.0231)	0.9006 (0.0132)	0.9064 (0.0122)	0.9107 (0.0144)
	maxmin est.	**0.0732** (0.0338)	**0.2037** (0.1014)	**0.4585** (0.1099)	**0.0152** (0.0028)	**0.0258** (0.0057)	**0.0552** (0.0181)

Table 6. Mean (and in brackets: standard deviation) of the L_2 error for the maxmin regression estimates, compared to L_2 error of neural networks (est. 1) and regression trees (est. 2). The regression function is m_8, m_9, m_{10} or m_{11}, respectively.