A DIFFERENCE OF CONVEX OPTIMIZATION ALGORITHM
FOR PIECEWISE LINEAR REGRESSION

ADIL BAGIROV* AND SONA TAHERI

Faculty of Science and Technology, Federation University Australia
Victoria, Australia

SOODABEH ASADI

Department of Mathematics, Shahrekord University, Iran

(Communicated by Gerhard-Wilhelm Weber)

Abstract. The problem of finding a continuous piecewise linear function approximating a regression function is considered. This problem is formulated as a nonconvex nonsmooth optimization problem where the objective function is represented as a difference of convex (DC) functions. Subdifferentials of DC components are computed and an algorithm is designed based on these subdifferentials to find piecewise linear functions. The algorithm is tested using some synthetic and real world data sets and compared with other regression algorithms.

1. Introduction. The aim of this paper is to develop an algorithm for nonparametric estimation of a multivariate regression function over the space of continuous piecewise linear functions.

Nonparametric methods are usually applied to estimate a regression function when a priori information about this function is not known. These methods minimize a kind of least squares risk of the regression estimate over a given function space. Such methods include regression trees like CART [10], adaptive spline fitting like MARS [13], artificial neural networks (ANN) [14] and Support Vector Machines (SVM) for regression [11, 23].

In papers [3, 5], a function space consisting of continuous piecewise linear functions is considered to estimate a regression function. The representation of these functions as a maximum of minimum of linear functions is used to formulate an empirical least squares risk. This risk function is nonconvex and nonsmooth. In [4], an algorithm based on the discrete gradient method from [6, 7] is developed to minimize the risk function. The discrete gradient method uses function values to approximate the subdifferential of the objective function and therefore, this method is time consuming for regression problems in large data sets.

In this paper, we introduce a new algorithm for the estimation of a regression function over the space of continuous piecewise linear functions. This algorithm
exploits an implicit difference of convex (DC) representation of the empirical risk function. The use of such representation allows one to efficiently calculate subgradients of DC components. We present computational results with synthetic and real world data sets and compare the proposed algorithm with other algorithms for regression.

The structure of the paper is as follows. The problem statement is given in Section 2. Subdifferentials of the empirical least squares risk function are described in Section 3. In Section 4, the minimization algorithm is introduced. Numerical results are reported in Section 5. Section 6 concludes the paper.

In what follows, we denote by $\mathbb{R}^n$ the $n$-dimensional Euclidean space with the inner product $\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i$ and the associated norm $\|u\| = (\langle u, u \rangle)^{1/2}$, $u, v \in \mathbb{R}^n$. Vectors are considered as a row.

$B_\varepsilon(x) = \{ y \in \mathbb{R}^n : \| y - x \| < \varepsilon \}$ is the open ball centered at $x$ with the radius $\varepsilon > 0$. “conv” denotes the convex hull of a set.

2. Problem statement. We consider the nonparametric regression estimation problem. The definition of this problem is given, for example, in [16]. The problem can be formulated as the minimization of the so-called $L_2$-risk over the set of all measurable functions. Usually, in an application the underlying distribution is not known and, therefore, the $L_2$-risk is estimated by an empirical $L_2$-risk [3, 4].

In regression analysis, an $\mathbb{R}^n \times \mathbb{R}$-valued random vector $(a, b)$ with $E b^2 < \infty$ is considered and the dependency of $b$ on the value of $a$ is of interest. More precisely, the goal is to find a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that $\varphi(a)$ is a “good approximation” of $b$. In applications, usually the set of data $A = \{(a_i, b_i) \in \mathbb{R}^n \times \mathbb{R} : i = 1, \ldots, m\}$ is given, where $a_i \in \mathbb{R}^n$ are values of $n$ input variables and $b_i \in \mathbb{R}$ is their output.

Then the empirical $L_2$-risk can be formulated as:

$$\frac{1}{m} \sum_{i=1}^{m} (\varphi(a_i) - b_i)^2.$$ (1)

For least squares estimates, first a class $\mathcal{F}$ of functions $\varphi : \mathbb{R}^n \to \mathbb{R}$ is chosen and then the estimate is defined by minimizing (1) over $\mathcal{F}$.

In this paper, $\mathcal{F}$ is defined as a set of continuous piecewise linear functions. It is known that such functions can be represented as a max-min of the finite number of linear functions [9, 15]. Therefore, all functions in $\mathcal{F}$ are given as a maximum of minimum of linear functions, and this representation is used to formulate the empirical $L_2$-risk function. More precisely, let $K \in \mathbb{N}$ and $M_1, \ldots, M_K \in \mathbb{N}$ be parameters of the estimate and set

$$\mathcal{F} = \left\{ \varphi : \mathbb{R}^n \to \mathbb{R} : \varphi(a) = \max_{k=1, \ldots, K} \min_{j=1, \ldots, M_k} \left( \langle a^{kj}, a \rangle + y_{kj} \right), a \in \mathbb{R}^n, \right. \\
\left. \text{for some } a^{kj} \in \mathbb{R}^n, y_{kj} \in \mathbb{R} \right\}.$$

Then the minimization of the empirical $L_2$-risk function, defined in (1), over the set $\mathcal{F}$ can be reformulated as:

$$\text{minimize } F(x, y) = \frac{1}{m} \sum_{i=1}^{m} \left( \max_{k=1, \ldots, K} \min_{j=1, \ldots, M_k} \left( \langle a^{kj}, a^i \rangle + y_{kj} \right) - b_i \right)^2,$$ (2)

subject to $a^{kj} \in \mathbb{R}^n$, $y_{kj} \in \mathbb{R}$, $j = 1, \ldots, M_k$, $k = 1, \ldots, K$. 

Here
\[x = (x_1^{11}, \ldots, x_n^{11}, \ldots, x_1^{KM}, \ldots, x_n^{KM}) \in \mathbb{R}^{nq}, \quad y = (y_1^{11}, \ldots, y_{KM}) \in \mathbb{R}^q,\]
and \(q = \sum_{k=1}^{K} M_k.\)

3. Subdifferentials of empirical \(L_2\)-risk function. In this section, first we show that the empirical \(L_2\)-risk function \(F\), defined in (2), is DC and then formulate subdifferentials of its DC components (details concerning DC functions can be found, for example, in [1,17,24,25]).

Consider the function
\[\psi_{ik}(x,y) = \min_{j=1,\ldots,M_k} \{\langle x^{kj}, a^i \rangle + y_{kj}\}, \quad i = 1, \ldots, m, \ k = 1, \ldots, K.\]

This function is DC [4]:
\[\psi_{ik}(x,y) = \psi_{ik}^1(x,y) - \psi_{ik}^2(x,y),\]
where
\[\psi_{ik}^1(x,y) = \sum_{j=1}^{M_k} (\langle x^{kj}, a^i \rangle + y_{kj}), \quad \psi_{ik}^2(x,y) = \max_{j=1,\ldots,M_k} \sum_{t=1, t \neq j}^{M_k} (\langle x^{kt}, a^i \rangle + y_{kt}).\]

The function \(\psi_{ik}^1\) is linear and the function \(\psi_{ik}^2\) is convex piecewise linear since it is represented as a maximum of linear functions. Next, consider the function:
\[\varphi_i(x,y) = \max_{k=1,\ldots,K} \psi_{ik}(x,y), \quad i = 1, \ldots, m.\]

This function is also DC [4]:
\[\varphi_i(x,y) = \varphi_{i1}(x,y) - \varphi_{i2}(x,y),\]
where
\[\varphi_{i1}(x,y) = \max_{k=1,\ldots,K} \left(\psi_{ik}^1(x,y) + \sum_{j=1, j \neq k}^{K} \psi_{ij}^2(x,y)\right), \quad \varphi_{i2}(x,y) = \sum_{j=1}^{K} \psi_{ij}^2(x,y).\]

Both functions \(\varphi_{i1}\) and \(\varphi_{i2}\) are convex piecewise linear.

**Proposition 1.** The empirical \(L_2\)-risk function \(F\) is DC.

**Proof.** The function \(F\) can be rewritten as:
\[F(x,y) = \frac{1}{m} \sum_{i=1}^{m} (\varphi_i(x,y) - b_i)^2.\]

Define the functions \(\phi_1 : \mathbb{R}^{nq} \times \mathbb{R}^q \to \mathbb{R}^m\) and \(\phi_2 : \mathbb{R}^m \to \mathbb{R}\) such that:
\[\phi_1(x,y) = (\varphi_1(x,y), \ldots, \varphi_m(x,y)), \quad \phi_2(z_1, \ldots, z_m) = \frac{1}{m} \sum_{i=1}^{m} (z_i - b_i)^2.\]

Since each component function \(\varphi_i\) is DC the function \(\phi_1\) is also DC. Moreover, the function \(\phi_2\) as a quadratic function is DC. It is obvious that \(F = \phi_2 \circ \phi_1.\) Then it follows from Proposition 3.4 from [25] that the function \(F\) is DC.

**Remark 1.** Although the function \(F\) is DC, in general, it is not easy to get its explicit DC representation. However, we show that subdifferentials of its DC components can be formulated explicitly.
Some properties of the function $F$ were studied in [4]. In particular, it is shown that this function is locally Lipschitz. This means that the function $F$ is subdifferentiable in $\mathbb{R}^{nq} \times \mathbb{R}^s$. Next, we formulate subdifferentials of DC components of this function. Let $I = \{1, \ldots, m\}$.

**Proposition 2.** Let $F_1$ and $F_2$ be DC components of the function $F$: $F(x, y) = F_1(x, y) - F_2(x, y)$, $(x, y) \in \mathbb{R}^{nq} \times \mathbb{R}^s$. Then subdifferentials of the functions $F_1$ and $F_2$ at $(x, y)$ can be expressed as:

$$
\partial F_1(x, y) = 2 \left[ \sum_{i \in I_+(x, y)} (\varphi_i(x, y) - b_i) \partial \varphi_{i1}(x, y) - \sum_{i \in I_-(x, y)} (\varphi_i(x, y) - b_i) \partial \varphi_{i2}(x, y) \right],
$$

(3)

$$
\partial F_2(x, y) = 2 \left[ \sum_{i \in I_+(x, y)} (\varphi_i(x, y) - b_i) \partial \varphi_{i1}(x, y) - \sum_{i \in I_-(x, y)} (\varphi_i(x, y) - b_i) \partial \varphi_{i2}(x, y) \right].
$$

(4)

Here $I_+(x, y) = \{i \in I : \varphi_i(x, y) > b_i\}$, $I_-(x, y) = \{i \in I : \varphi_i(x, y) < b_i\}$.

**Proof.** The function $F$ can be represented as a composition of functions $\phi_1$ and $\phi_2$:

$$
F(x, y) = \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y)).
$$

Since functions $F_1$ and $F_2$ are convex on $\mathbb{R}^{nq} \times \mathbb{R}^s$ they are directionally differentiable. Their directional derivatives at a point $(x, y) \in \mathbb{R}^{nq} \times \mathbb{R}^s$ in a direction $d = (d_x, d_y) \in \mathbb{R}^{nq} \times \mathbb{R}^s$ can be computed as:

$$
F'_1((x, y), d) = \sum_{i \in I_+(x, y)} \frac{\partial \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y))}{\partial z_i} \varphi'_{i1}((x, y), d) - \sum_{i \in I_-(x, y)} \frac{\partial \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y))}{\partial z_i} \varphi'_{i2}((x, y), d),
$$

and

$$
F'_2((x, y), d) = \sum_{i \in I_+(x, y)} \frac{\partial \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y))}{\partial z_i} \varphi'_{i2}((x, y), d) - \sum_{i \in I_-(x, y)} \frac{\partial \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y))}{\partial z_i} \varphi'_{i1}((x, y), d).
$$

Here $\partial \phi_2(z_1, \ldots, z_m)/\partial z_i$ is the partial derivative of the function $\phi_2$ with respect to the variable $z_i$, $i \in I$. It is obvious that

$$
\frac{\partial \phi_2(\varphi_1(x, y), \ldots, \varphi_m(x, y))}{\partial z_i} = 2(\varphi_i(x, y) - b_i), \; i = 1, \ldots, m.
$$

Then expressions (3) and (4) for subdifferentials $\partial F_1(x, y)$ and $\partial F_2(x, y)$ follow directly from corresponding expressions for directional derivatives $F'_1((x, y), d)$ and $F'_2((x, y), d)$. \qed

**Remark 2.** Since subdifferentials of functions $\varphi_{i1}$ and $\varphi_{i2}$ are polytopes at any $(x, y) \in \mathbb{R}^{nq} \times \mathbb{R}^s$ expressions (3) and (4) imply that subdifferentials $\partial F_1(x, y)$ and $\partial F_2(x, y)$ are also polytopes at any $(x, y) \in \mathbb{R}^{nq} \times \mathbb{R}^s$. 
It follows from optimality conditions for general unconstrained nonsmooth DC optimization problems [7, 12] that for a point \((x^*, y^*) \in \mathbb{R}^{nq} \times \mathbb{R}^q\) to be a local minimizer of the problem (2) it is necessary that
\[
\partial F_2(x^*, y^*) \subseteq \partial F_1(x^*, y^*). \tag{5}
\]
However, in practice it is not easy to verify this condition as it requires the calculation of the whole subdifferential \(\partial F_2(x^*, y^*)\). Therefore, the necessary condition (5) is replaced with the more relaxed condition of criticality:
\[
\partial F_1(x^*, y^*) \cap \partial F_2(x^*, y^*) \neq \emptyset. \tag{6}
\]
Most existing algorithms guarantee convergence to critical points satisfying the condition (6).

**Remark 3.** Problem (2) is a global optimization problem. It becomes a large scale problem if the number of linear functions used to estimate a regression function is relatively large. In this case, most global optimization algorithms are not efficient for solving Problem (2), and local search algorithms are the only choice. Moreover, local search algorithms can be used to design efficient global optimization methods for solving Problem (2).

**Remark 4.** As it is noted in Remark 1, in general, we do not have an explicit DC representation of the function \(F\). However, the explicit expressions for subdifferentials of its DC components are available. Most existing algorithms for local DC optimization use not only subgradients but also values of DC components. These algorithms include, for example, the DCA algorithm [24] and the version of the proximal bundle method for DC optimization [18]. Such algorithms cannot be used to solve Problem (2) since the values of DC components are not always available. Therefore, in the next section we design an algorithm which does not require the explicit DC representation of the function \(F\). More specifically, the proposed algorithm does not use the values of DC components, it uses only values of the function \(F\) and subgradients of its DC components.

**4. Numerical algorithm.** In this section, we design an algorithm for solving Problem (2). The algorithm starts from any initial point and finds a critical point of this problem.

For convenience, we consider more general form of Problem (2):
\[
\minimize \ f(x) \ \text{subject to} \ x \in \mathbb{R}^n, \tag{7}
\]
where \(f(x) = f_1(x) - f_2(x)\) and functions \(f_1\) and \(f_2\) are convex on \(\mathbb{R}^n\). Moreover, we assume that \(\text{dom} f_1 = \text{dom} f_2 = \mathbb{R}^n\), subdifferentials \(\partial f_1(x)\) and \(\partial f_2(x)\) are polytopes at any \(x \in \mathbb{R}^n\) and
\[
f_* = \inf \{ f(x) : x \in \mathbb{R}^n \} \geq 0. \tag{8}
\]
Notice that the objective function in Problem (2) satisfies these conditions.

Take any \(\lambda > 0\) and define the following set at a point \(x \in \mathbb{R}^n\):
\[
Q_1(x, \lambda) = \text{conv} \{ u \in \partial f_1(x + \lambda d) : d \in S_1 \}.
\]
Here \(S_1 = \{ d \in \mathbb{R}^n : \|d\| = 1 \}\) is the unit sphere. It is obvious that the set \(Q_1(x, \lambda)\) is convex and compact for any \(x \in \mathbb{R}^n\) and \(\lambda > 0\).

Denote by \(Q_2(x)\) the set of vertices of the subdifferential \(\partial f_2(x), x \in \mathbb{R}^n\). For a given \(v \in Q_2(x)\) define the following set
\[
\bar{Q}_v(x, \lambda) = Q_1(x, \lambda) - v.
\]
Convexity of functions \( f_1 \) and \( f_2 \) implies that for any \( w \in \bar{Q}_v(x, \lambda) \) and \( v \in Q_2(x) \)

\[
f(x + \lambda d) - f(x) \leq \lambda \langle w, d \rangle \quad \forall d \in \mathbb{R}^n. \tag{9}
\]

In turn this implies that

\[
f(x + \lambda d) - f(x) \leq \lambda \max_{w \in \bar{Q}_v(x, \lambda)} \langle w, d \rangle \quad \forall d \in \mathbb{R}^n.
\]

Let \( \lambda, \delta > 0 \) be given numbers.

**Definition 4.1.** A point \( x^* \in \mathbb{R}^n \) is called a \((\lambda, \delta)\)-critical point of Problem (7) iff:

\[
\partial f_2(x^*) \cap (Q_1(x^*, \lambda) + B_\delta(0_n)) \neq \emptyset. \tag{11}
\]

If \( \lambda = \delta = 0 \) then we get the definition of critical points.

Assume that a point \( x \in \mathbb{R}^n \) is not a \((\lambda, \delta)\)-critical point of Problem (7). Then \( \|v - u\| \geq \delta \) for all \( v \in Q_2(x) \) and \( u \in Q_1(x, \lambda) \). Let \( \bar{u} \in Q_1(x, \lambda) \) and \( \bar{v} \in Q_2(x) \) are such that

\[
\|\bar{u} - \bar{v}\| = \max_{u \in Q_1(x, \lambda)} \min_{v \in Q_2(x)} \|u - v\| \geq \delta.
\]

**Proposition 3.** Assume that the point \( x \in \mathbb{R}^n \) is not a \((\lambda, \delta)\)-critical point of Problem (7). Then the direction \( d^0 = -\|w^0\|^{-1}w^0 \), where \( w^0 = \bar{u} - \bar{v} \), is a descent direction of the function \( f \) at \( x \) and

\[
f(x + \lambda d^0) - f(x) \leq -\lambda \|w^0\|. \tag{12}
\]

**Proof.** From the necessary condition for a minimum we have

\[
\langle w^0, w - w^0 \rangle \geq 0 \quad \forall w \in \bar{Q}_v(x, \lambda)
\]

or \( \langle w^0, w \rangle \geq \|w^0\|^2 \quad \forall w \in \bar{Q}_v(x, \lambda) \). Dividing both sides by \( -\|w^0\| \) we have \( \langle w, d^0 \rangle \leq -\|w^0\| \quad \forall w \in \bar{Q}_v(x, \lambda) \). This together with (10) imply that \( d^0 \) is a descent direction satisfying (12). \( \square \)

It follows from Proposition 3 that if the point \( x \) is not a \((\lambda, \delta)\)-critical then the sets \( Q_1(x, \lambda) \) and \( Q_2(x) \) can be used to find a direction of sufficient decrease of the function \( f \) at \( x \). However, the computation of these sets is not always possible. Next, we design an algorithm which uses a finite number of elements from these two sets to compute descent directions.

Let \( \lambda > 0 \) be a given number and \( x \in \mathbb{R}^n \) be a given point.
Algorithm 1 Computation of descent directions.

1: (Initialization). Select a search control parameter \( c \in (0,1) \) and the initial direction \( d^0 \in S_1 \). Compute the subgradients \( u^1 \in \partial f_1(x + \lambda d^1) \) and \( v^1 \in \partial f_2(x) \) such that \( f'_2(x, d^1) = (v^1, d^1) \). Set \( Q_1 := \{u^1\}, Q_2 := \{v^1\} \) and \( j := 1 \).

2: (Computation of maximum deviation subgradients). Compute \( \bar{u}^j \in Q_1^j \) and \( \bar{v}^j \in Q_2^j \) such that

\[
\|\bar{v}^j - \bar{u}^j\|^2 = \max_{v \in Q_2^j} \min_{u \in Q_1^j} \|v - u\|^2
\]

and define \( w^j = \bar{u}^j - \bar{v}^j \).

3: (Stopping criterion). If

\[
\|w^j\| \leq \delta,
\]

then STOP. \( x \) is a \((\lambda, \delta)\)-critical point.

4: (Computation of a search direction). Compute \( d^{j+1} = -\|w^j\|^{-1} w^j \). If

\[
f(x + \lambda d^{j+1}) - f(x) \leq -c\|w^j\|
\]

then STOP. The direction \( d^{j+1} \) is a direction of sufficient decrease.

5: (Computation of new subgradients). Compute the subgradients \( u^{j+1} \in \partial f_1(x + \lambda d^{j+1}) \), \( v^{j+1} \in \partial f_2(x) \) such that \( f'_2(x, d^{j+1}) = (v^{j+1}, d^{j+1}) \) and the sets \( Q_1^{j+1} := \text{conv} \{Q_1^j \cup \{u^{j+1}\}\} \), \( Q_2^{j+1} := \text{conv} \{Q_2^j \cup \{v^{j+1}\}\} \). Set \( j := j + 1 \) and go to Step 2.

Remark 5. One can use the following scheme to compute subgradients \( v \in \partial f_2(x) \) such that \( f'_2(x, d) = \langle v, d \rangle \) for \( d \in S_1 \). Let \( \alpha > 0 \) be a sufficiently small number. Compute a subgradient \( \bar{v} \in \partial f_2(x + \alpha d) \). Then semismoothness of the function \( f_2 \) implies that \( f'_2(x, d) \approx \langle \bar{v}, d \rangle \) and the subgradient \( v \) can be replaced by the subgradient \( \bar{v} \).

Next, we prove that Algorithm 1 is finite convergent.

Proposition 4. If stopping criteria (13) and (14) in Algorithm 1 do not hold at the \( j \)-th iteration then \( (w^{j+1}, v^{j+1}) \notin [Q_1^j, Q_2^j] \).

Proof. If the algorithm does not stop at the \( j \)-th iteration, \( j \geq 1 \), then

\[
\|w^j\| > \delta,
\]

and

\[
f(x + \lambda d^{j+1}) - f(x) \geq -c\|w^j\|.
\]

According to Step 2

\[
\bar{u}^j = \arg\min_{u \in Q_1^j} \|u - \bar{v}^j\|^2.
\]

From the necessary condition for a minimum we have \( \langle \bar{u}^j - \bar{v}^j, u - \bar{u}^j \rangle \geq 0 \ \forall u \in Q_1^j \) or \( \langle \bar{u}^j - \bar{v}^j, u - \bar{v}^j \rangle \geq \|\bar{u}^j - \bar{v}^j\|^2 \). This implies that

\[
\langle w^j, u - \bar{v}^j \rangle \geq \|w^j\|^2 \ \forall u \in Q_1^j.
\]

Dividing both sides by \( -\|w^j\| \) we have

\[
\langle u - \bar{v}^j, d^{j+1} \rangle \leq -\|w^j\| \ \forall u \in Q_1^j.
\]

It follows from (9) that \( f(x + \lambda d^{j+1}) - f(x) \leq \lambda (w^{j+1} - v^{j+1}, d^{j+1}) \). Then from (16) we get

\[
\langle w^{j+1} - v^{j+1}, d^{j+1} \rangle > -c\|w^j\|.
\]
Let
\[ \sigma_i^j(d) = \max_{z \in Q^j_i} \{z,d\} \]
be a support function of the set \( \bar{\mathcal{Q}}^j_i \), \( i = 1,2 \). Consider
\[ p_j = \sigma_1^j(d^{j+1}) - \sigma_2^j(d^{j+1}), \quad \bar{p}_j = \sigma_1^{j+1}(d^{j+1}) - \sigma_2^{j+1}(d^{j+1}). \]

Since \( u^{j+1} \in \bar{Q}_1^{j+1}, \ v^{j+1} \in \bar{Q}_2^{j+1} \) and
\[ \langle v^{j+1}, d^{j+1} \rangle = f'(x,d^{j+1}) = \max_{v \in Q_2^{j+1}} \langle v, d^{j+1} \rangle = \sigma_2^{j+1}(d^{j+1}) \]
we get \( \bar{p}_j \geq (u^{j+1} - v^{j+1}, d^{j+1}) \). Then it follows from (18) that
\[ \bar{p}_j \geq -c\|w^j\|. \quad (19) \]
Furthermore, from the definition of the direction \( d^{j+1} \) we have \( p_j = -\|w^j\| \). Therefore,
\[ \bar{p}_j - p_j > (1 - c)\|w^j\|. \quad (20) \]

Since \( c \in (0,1) \), \( \bar{Q}_1^j \subseteq \bar{Q}_1^{j+1} \) and \( \bar{Q}_2^j \subseteq \bar{Q}_2^{j+1} \) it follows from (20) that at least either \( \bar{Q}_1^{j+1} \neq \bar{Q}_1^j \) or \( \bar{Q}_2^{j+1} \neq \bar{Q}_2^j \). This means that \( (u^{j+1}, v^{j+1}) \notin [\bar{Q}_1^j, \bar{Q}_2^j] \). In addition, \( \bar{Q}_1^{j+1} \neq \bar{Q}_1^j \) for any \( j \geq 1 \). Indeed, if \( \bar{Q}_1^{j+1} = \bar{Q}_1^j \) for some \( j \geq 1 \) then \( \bar{Q}_2^j \subseteq \bar{Q}_2^{j+1} \). This implies that \( \bar{p}_j \leq p_j \) which contradicts (20).

**Proposition 5.** Algorithm 1 terminates after a finite number of iterations.

**Proof.** Assume the contrary, that is Algorithm 1 generates an infinite sequence of pairs of the sets \( [\bar{Q}_1^j, \bar{Q}_2^j] \). It is obvious that for any \( j \geq 1 \)
\[ \bar{Q}_1^j \subset \bar{Q}_1^{j+1}, \bar{Q}_2^j \subset \bar{Q}_2^{j+1}, \bar{Q}_1^j \subset Q_1(x,\lambda), \bar{Q}_2^j \subset \partial f_2(x). \quad (21) \]

Moreover, conditions (13) and (14) do not hold for any \( j \geq 1 \).

The sets \( \bar{Q}_1^j, Q_1(x,\lambda) \) and \( \partial f_2(x) \) are convex and compact. Furthermore, the set \( \bar{Q}_2^j \) is finite. Take any \( d \in \mathbb{R}^n, \ d \neq 0_n \). Then the first two inclusions from (21) imply that sequences \{\( \sigma_i^j(d) \)\}, \( i = 1,2 \) are nondecreasing. Using the last two inclusions in (21) we can conclude that these sequences are bounded above. Then they are convergent, and for any \( \varepsilon > 0 \) there exists \( j_\varepsilon > 1 \) such that
\[ 0 \leq \sigma_i^{j+1}(d) - \sigma_i^j(d) < \varepsilon, \ i = 1,2 \]
for all \( j > j_\varepsilon \). Since the set \( S_1 \) is compact the inequalities (22) are true for any \( d \in S_1 \). Taking into account the condition (13) we get from (20) that
\[ \bar{p}_j - p_j > (1 - c)\delta \ \forall j > 0. \quad (23) \]

In (22) by taking \( \varepsilon = (1 - c)\delta/2 \) for \( d = d^{j+1} \) we get
\[ \sigma_i^{j+1}(d^{j+1}) - \sigma_i^j(d^{j+1}) < (1 - c)\delta/2 \ \forall j > j_\varepsilon, \ i = 1,2. \]

Then
\[ \bar{p}_j - p_j \leq \sigma_i^{j+1}(d^{j+1}) - \sigma_i^j(d^{j+1}) < (1 - c)\delta/2 \]
which contradicts (23), and this completes the proof. \qed
Next, we describe an algorithm for finding \((\lambda, \delta)\)-critical points for given \(\lambda > 0, \delta > 0\).

**Algorithm 2** Finding \((\lambda, \delta)\)-critical points.

1. (Initialization). Select any starting point \(x^1 \in \mathbb{R}^n\), numbers \(c_1 \in (0, 1)\) and \(c_2 \in (0, 1]\). Set \(j := 1\).
2. (Finding search direction) Apply Algorithm 1 with \(c = c_1\) to find a search direction at the point \(x^j\). This algorithm terminates after a finite number of iterations \(l_j > 0\) with either \(\|w^j\| \leq \delta\) or finds a descent direction \(d^j \in S_1\) such that
   \[
   f(x^j + \lambda d^j) - f(x^j) \leq -c_1 \lambda \|w^j\|.
   \]
3. (Stopping criterion) If \(\|w^j\| \leq \delta\), then STOP. \(x^j\) is a \((\lambda, \delta)\)-critical point.
4. (Line search) If (24) holds then find the step-length \(\alpha_j > 0\) as follows:
   \[
   \alpha_j = \arg\max \left\{ \alpha \geq 0 : f(x^j + \alpha d^j) - f(x^j) \leq -c_2 \alpha \|w^j\| \right\}.
   \]
5. (Updating iteration) Set \(x^{j+1} := x^j + \alpha_j d^j, j := j + 1\) and go to Step 2.

**Proposition 6.** Assume that the condition (8) holds. Then Algorithm 2 finds a \((\lambda, \delta)\)-critical point of Problem (7) in at most \(j_1\) iterations, where
   \[
   j_1 = \left\lceil \frac{f(x^1)}{c_2 \lambda \delta} \right\rceil. \tag{25}
   \]

**Proof.** Assume the contrary, that is the sequence \(\{x^j\}\) is infinite and points \(x^j\) are not \((\lambda, \delta)\)-critical for all \(j = 1, 2, \ldots\). This means that \(\|w^j\| \geq \delta, j = 1, 2, \ldots\). Since \(c_2 \leq c_1\) it follows from (24) that \(\alpha_j \geq \lambda\) for any \(j > 0\). Then \(f(x^{j+1}) - f(x^j) \leq -c_2 \lambda \delta\), and therefore
   \[
   f(x^{j+1}) - f(x^1) \leq -c_2 j \lambda \delta,
   \]
which means that \(f(x^j) \to -\infty\) as \(j \to \infty\). This contradicts the condition \(f_* \geq 0\) and therefore the algorithm is finite convergent. Since \(f_* \leq f(x^{j+1})\) it is obvious that the maximum number of iterations \(j_1\) is given by (25). \(\square\)

Now we are ready to design an algorithm to find critical points of Problem (7). Let \(\{\lambda_j\}\) and \(\{\delta_j\}\) be sequences such that \(\lambda_j, \delta_j \downarrow 0\) as \(j \to \infty\).

**Algorithm 3** Finding critical points.

1. (Initialization). Select any starting point \(x^1 \in \mathbb{R}^n\) and an optimality tolerance \(\varepsilon > 0\). Set \(j := 1\).
2. Apply Algorithm 2 starting from the point \(x^j\) to find the \((\lambda_j, \delta_j)\)-critical point \(x^{j+1}\).
3. (Stopping criterion) If \(\lambda_j < \varepsilon\) and \(\delta_j < \varepsilon\) then STOP. Otherwise, set \(j := j + 1\) and go to Step 2.

**Proposition 7.** Assume that the condition (8) holds, the level set \(\mathcal{J}(x^1) = \{x \in \mathbb{R}^n : f(x) \leq f(x^1)\}\) is bounded and \(\varepsilon = 0\). Then all limit points of the sequence \(\{x^j\}\) generated by Algorithm 3 are critical points of Problem (7).
Proof. Algorithm 2 is a descent algorithm, therefore, $x^j \in J(x^1)$ for all $j \geq 1$. Since the set $J(x^1)$ is compact the sequence $\{x^j\}$ has at least one limit point. Assume, without loss of generality, that $x^j \to \bar{x}$ as $j \to \infty$. For each $x^j$ one has
\[ \partial f_2(x^j) \cap (Q_1(x^j, \lambda_j) + B_{\delta_j}(0_n)) \neq \emptyset. \]
This implies that there is a sequence $\{z^j\}$ such that $z^j \in \partial f_2(x^j)$ and $z^j \in Q_1(x^j, \lambda_j) + B_{\delta_j}(0_n)$ for all $j \geq 1$.

It follows from the upper semicontinuity of the subdifferential maps $x \mapsto \partial f_i(x)$, $i = 1, 2$ that for any $\gamma > 0$ there exists $j_1 = j_1(\gamma) > 0$ such that
\[ \partial f_i(x^j) \subset \partial f_i(\bar{x}) + B_{\gamma}(0_n), \quad i = 1, 2 \]
for all $j > j_1$. Then according to the definition of the map $x \mapsto Q_1(x, \lambda)$ we get that for $\gamma > 0$ there exists $j_2 \geq j_1$ such that
\[ Q_1(x^j, \lambda_j) \subset \partial f_1(\bar{x}) + B_{\gamma}(0_n) \]
for all $j > j_2$. This along with (26) imply that
\[ z^j \in (\partial f_1(\bar{x}) + B_{\gamma + \delta_j}(0_n)) \cap (\partial f_2(\bar{x}) + B_{\gamma}(0_n)) \]
for all $j > j_2$. It is obvious that the sequence $\{z^j\}$ is bounded and, therefore, has at least one limit point. Let $\bar{z}$ be a limit point of the sequence $\{z^j\}$. Since $\gamma > 0$ is arbitrary and $\delta_j \downarrow 0$ as $j \to \infty$ it follows from (27) that $\bar{z} \in \partial f_1(\bar{x}) \cap \partial f_2(\bar{x})$. This means that the point $\bar{x}$ is a critical point of Problem (7).

Remark 6. There are some similarities between Algorithm 3 and an optimization algorithm introduced in [8]. Both algorithms are designed to solve nonsmooth unconstrained DC optimization problems, and they use the same sets to approximate the subdifferential of the first DC component. But there are also significant differences between these two algorithms. The algorithm from [8] uses only one subgradient of the second DC component at each iteration whereas Algorithm 3 uses a subset of the subdifferential of this function. In addition, in Algorithm 3 search directions are found by calculating the distance between two polytopes approximating subdifferentials of functions $f_1$ and $f_2$. However, in the algorithm from [8] these directions are computed using the set defined as the difference of the set, approximating the subdifferential of $f_1$, and one subgradient of the function $f_2$. The use of more than one subgradient from the subdifferential of the second DC component leads to the design of algorithms which can find solutions with a better quality than algorithms which use only one subgradient. This is especially important for solving nonconvex nonsmooth optimization problems.

Remark 7. The regression algorithm proposed in this paper can be considered as a generalization of the multiple linear regression (MLR) algorithm. If $K = 1$ and $M_1 = 1$ then the proposed algorithm becomes a version of the MLR algorithm.

5. Computational results. In this section, using synthetic and real world data sets we demonstrate the performance of the proposed piecewise linear regression algorithm (PWLREG) and compare it with other regression algorithms.

Synthetic data sets are used to:
1. show how the proposed algorithm works;
2. demonstrate the generalization ability of the PWLREG algorithm;
3. study the performance of the PWLREG algorithm depending on the level of noise;
4. compare the PWLREG algorithm with the piecewise linear regression algorithm (DGMREG) introduced in [4].

The synthetic data sets are generated using the following relationship between the input $U$ and the output $V$:

$$ V = s(U) + \sigma \gamma, $$

where $s$ is a regression function, $\gamma$ is standard normally distributed and independent of $U$ and $\sigma \geq 0$. $U$ is uniformly distributed on $[-2, 2]^n$. For the noise level (standard deviation) $\sigma$ we use three different values: 0, 0.5 and 1. Regression functions $s$ used to generate synthetic data in different spaces are given in [4,5].

Real world data sets are used to study the prediction ability of the PWLREG algorithm and to compare it with other well known regression algorithms such as the MLR, the linear SVM (SVM(Lin)) for regression, the SVM regression algorithm with the radial basis kernel function (SVM(RBF)), the adaptive spline fitting (MARS) algorithm and the artificial neural networks (ANN).

5.1. Implementation of algorithms. The PWLREG algorithm is based on Algorithm 3 which involves two other algorithms: Algorithms 1 and 2. Search directions are found by applying Algorithm 1. Most important step in this algorithm is Step 2 where search directions are computed by solving several quadratic programming subproblems. These problems are solved using a finite convergent algorithm introduced in [26].

A search control parameter $c$ in Algorithm 1 is chosen as $c = 0.2$. Algorithm 2 finds $(\lambda, \delta)$-stationary points for given $\lambda, \delta > 0$. The line search in Step 4 of this algorithm is performed using the backtracking method. Parameters $c_1$ and $c_2$ in Algorithm 2 are chosen as follows: $c_1 = c = 0.2$ and $c_2 = 0.01$. The optimality tolerance $\varepsilon$ in Algorithm 3 is set to $\varepsilon = 10^{-4}$. The sequences $\{\lambda_j\}$ and $\{\delta_j\}$ in this algorithm are chosen as follows: $\lambda_1 = 1$, $\lambda_{j+1} = 0.1\lambda_j$, $\delta_j = 10^{-6}$, $j \geq 1$.

All prediction methods are applied without testing their underlying assumptions. This means that some of these methods may not perform well as not all their underlying assumptions are met. For example, in many real life applications the relationship between the dependent and independent variables is highly nonlinear and the errors usually exhibit non-normality. These violations of assumptions may impact the prediction performance of the MLR.

Prediction performance of algorithms are estimated using five performance measures: the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), the Mean Squared Error (MSE), the Nash–Sutcliffe Coefficient of Efficiency (CE) [21] and the Pearson correlation coefficient ($r$).

Definitions of these measures are given below. Assume that $e_1, \ldots, e_m$, $m \geq 1$ are observed values for some parameter $e$ and $\bar{e}_1, \ldots, \bar{e}_m$ are their forecast values. Then

$$ RMSE = \left( \frac{1}{m} \sum_{i=1}^{m} (\bar{e}_i - e_i)^2 \right)^{1/2}; $$

$$ MAE = \frac{1}{m} \sum_{i=1}^{m} |\bar{e}_i - e_i|; $$

$$ MSE = \frac{1}{m - p} \sum_{i=1}^{m} (\bar{e}_i - e_i)^2; $$
\[ CE = 1 - \frac{\sum_{i=1}^{m} (e_i - \bar{e}_i)^2}{\sum_{i=1}^{m} (e_i - e_0)^2} \].

In the MSE measure, \( p \) stands for the number of parameters in the model. In the CE measure, \( e_0 \) is the mean of observed values. The small values of RMSE, MAE and MSE measures indicate small deviations of the predictions from actual observations. The CE ranges from \(-\infty\) to 1. An efficiency \( CE = 1 \) means a perfect prediction. An efficiency of 0 indicates that the model predictions are as accurate as the mean of the observed data and an efficiency \(-\infty < CE < 0 \) occurs when the observed mean is a better predictor than the model.

PWLREG and DGMREG algorithms were implemented in Fortran 95 and compiled using free compiler gfortran. We use R implementations of the MLR, SVM(Lin), SVM(RBF), MARS and ANN regression algorithms. Parameters in the SVM regression, the MARS and the ANN algorithms are chosen according to [19, 20, 22], respectively. Computational experiments were carried out on a 2.50 GHz Intel(R) Core(TM) i5 machine with 8 GB of RAM.

5.2. Illustrative example. In this subsection using a synthetic data set, we demonstrate how the proposed algorithm works. The synthetic data set contains 1000 observations and one input variable, that is we have \((a^i, b_i)\), \( i = 1, \ldots, 1000 \), where \( a^i \in \mathbb{R} \) and \( b_i \in \mathbb{R} \). This data set is generated using the regression function

\[ s(u) = 2 \max (1, \min(2u + 3, -8u + 3)), \]  \hspace{1cm} (28)

and the level of noise \( \sigma = 0 \).

In the implementation of the PWLREG (also DGMREG) algorithm it is important to choose the right number \( K \) of minimum functions under maximum and the number \( M_k \), \( k = 1, \ldots, K \) of linear functions under each minimum function. The numbers \( M_k \) can vary depending on \( k \in \{1, \ldots, K\} \), however in our calculations we choose them the same for all \( k \). Thus, we use the notation \((K, M)\) to give the configuration of the max-min functions. Here \( K \) stands for the number of minimum functions under maximum and \( M \) stands for the number of linear functions under each minimum function. We consider the following configurations: \((1, 1), (2, 1), (2, 2), (3, 2), (4, 2), (5, 2)\). Piecewise linear functions with larger values of \( K \) and \( M \) are not considered as their use may lead to overfitting.

We explain the proposed approach for one of these configurations. Consider the piecewise linear function with the configuration \((2, 2)\) that is \( K = 2 \) and \( M_1 = M_2 = 2 \). In this case the objective function \( F \) of Problem (2) is defined as:

\[ F(x, y) = \frac{1}{1000} \sum_{i=1}^{1000} \left( \max \left( \min \{x^{11}a^i + y_{11}, x^{12}a^i + y_{12}\}, \min \{x^{21}a^i + y_{21}, x^{22}a^i + y_{22}\} \right) - b_i \right)^2. \]

Here \( x = (x^{11}, x^{12}, x^{21}, x^{22}) \in \mathbb{R}^4 \) and \( y = (y_{11}, y_{12}, y_{21}, y_{22}) \in \mathbb{R}^4 \). Functions \( \psi_{i1} \) and \( \psi_{i2} \), \( i = 1, \ldots, 1000 \) are as follows:

\[ \psi_{i1}(x, y) = \min \{x^{11}a^i + y_{11}, x^{12}a^i + y_{12}\}, \quad \psi_{i2}(x, y) = \min \{x^{21}a^i + y_{21}, x^{22}a^i + y_{22}\} \]

Their DC representations are:

\[ \psi_{i1}(x, y) = \psi_{i1}^1(x, y) - \psi_{i1}^2(x, y), \quad \psi_{i2}(x, y) = \psi_{i2}^1(x, y) - \psi_{i2}^2(x, y), \]

where

\[ \psi_{i1}^1(x, y) = (x^{11} + x^{12})a^i + (y_{11} + y_{12}), \quad \psi_{i1}^2(x, y) = \max \{x^{11}a^i + y_{11}, x^{12}a^i + y_{12}\}, \]

\[ \psi_{i2}^1(x, y) = (x^{21} + x^{22})a^i + (y_{21} + y_{22}), \quad \psi_{i2}^2(x, y) = \max \{x^{21}a^i + y_{21}, x^{22}a^i + y_{22}\}. \]
The functions $\varphi_i$, $i = 1, \ldots, 1000$ are given as:

$$\varphi_i(x, y) = \max \{ \psi_{i1}(x, y), \psi_{i2}(x, y) \}$$

and their DC representations are:

$$\varphi_i(x, y) = \varphi_{i1}(x, y) - \varphi_{i2}(x, y),$$

where

$$\varphi_{i1}(x, y) = \max(\psi_{11}^1(x, y) + \psi_{22}^1(x, y), \psi_{11}^2(x, y) + \psi_{22}^2(x, y)), \quad \varphi_{i2}(x, y) = \psi_{11}^2(x, y) + \psi_{22}^2(x, y).$$

Since the functions $\varphi_{i1}$ and $\varphi_{i2}$ are convex their subgradients can be computed applying the subdifferential calculus. Using these subgradients one can compute subgradients of the functions $F_1$ and $F_2$ by applying formulas (3) and (4), respectively. Algorithm 3 is applied to solve Problem (2) using subdifferentials of the functions $F_1$ and $F_2$ and to find piecewise linear functions for each configuration separately.

In order to determine the best configuration of the piecewise linear function the data set is divided into two parts: the training set containing 800 (80% of all) instances and the validation set containing the rest 200 (20% of all) instances. The training set is used to compute coefficients of linear functions using above mentioned six configurations of piecewise linear functions. Then the validation set is used to identify the best configuration using the RMSE as a performance measure. More specifically, the piecewise linear function providing the smallest value of the RMSE measure on the validation set is accepted as the estimation to the regression function.

The steps of the PWLREG algorithm for finding the best configuration are illustrated in Figure 1. In this figure, blue dots represent the simulation data and red lines represent the estimation functions. For each configuration we present the values of RMSE obtained using the validation set. One can see that the piecewise linear function with the configuration $(3, 2)$ gives the smallest value of RMSE. Therefore, the PWLREG algorithm chooses this function as an estimation for the regression function.

### 5.3. Generalization ability of PWLREG algorithm

In this subsection synthetic data sets, containing from 500 to 5000 instances and from 1 to 10 input variables, are used to study the generalization ability of the proposed algorithm and to determine the CPU time required by this algorithm for training. These data sets are generated using regression functions given in [4, 5] with the standard deviation $\sigma = 1$.

Data sets are divided into training and test sets. In all cases, 60% of the data are used for training and 40% for testing. Results are reported in Table 1 where we present the values of RMSE for both training and test sets as well as the CPU time (in seconds) required for training. These results show that overall the proposed algorithm demonstrates a good generalization ability, especially in data sets with the number of instances $m = 3000$ and $m = 5000$. In these data sets, the difference between RMSE values for training and test sets is small. In small data sets with the number of instances $m \leq 1000$ this difference in some cases is quite large. These results show that the generalization ability of the PWLREG algorithm depends on the number of linear functions and the number of instances in a data set. In addition, the generalization ability of this algorithm depends on the number of input variables. In small data sets, the number of linear functions should be restricted to have a good generalization of the algorithm. However, this number can be increased as the...
the number of instances of a data set increases. One can see that the CPU time used by the algorithm for training is reasonable in all cases. Moreover, the increase of the size of data sets does not lead to a significant increase of computational effort required by the algorithm.

The graphical illustration of results by the PWLREG algorithm and their comparison with those obtained by SVM(Lin) and SVM(RBF) algorithms are presented in Figures 2, 3, 4 and 5. In Figures 2 and 3 data sets with one input variable are used. These data sets contain 500 and 5000 instances and are obtained using the regression function:

\[ s(u) = \begin{cases} 
1, & u \leq 0, \\
3, & \text{otherwise},
\end{cases} \]
Table 1. RMSE results and CPU time for PWLREG on synthetic data sets

| $m \cdot 10^n$ | $n$ | Train | Test | CPU | $m \cdot 10^n$ | $n$ | Train | Test | CPU |
|----------------|-----|-------|------|-----|----------------|-----|-------|------|-----|
| 5              | 1   | 1.089 | 1.084 | 0.09| 3              | 1   | 1.138 | 1.113 | 0.37|
| 5              | 2   | 0.890 | 0.856 | 0.14| 3              | 2   | 0.596 | 0.587 | 5.90|
| 5              | 3   | 0.982 | 1.146 | 0.87| 3              | 3   | 1.063 | 1.101 | 3.37|
| 5              | 4   | 1.143 | 1.395 | 0.59| 3              | 4   | 1.168 | 1.289 | 4.32|
| 5              | 7   | 1.307 | 1.409 | 0.39| 3              | 7   | 1.310 | 1.353 | 2.12|
| 5              | 10  | 1.311 | 1.406 | 0.28| 3              | 10  | 1.314 | 1.355 | 1.64|
| 1              | 1   | 1.061 | 1.101 | 0.14| 5              | 1   | 1.120 | 1.118 | 0.62|
| 1              | 2   | 0.665 | 0.725 | 1.70| 5              | 2   | 0.706 | 0.692 | 9.10|
| 1              | 3   | 1.003 | 1.077 | 1.22| 5              | 3   | 1.067 | 1.038 | 6.96|
| 1              | 4   | 1.312 | 1.225 | 0.59| 5              | 4   | 1.263 | 1.271 | 7.07|
| 1              | 7   | 1.368 | 1.257 | 0.59| 5              | 7   | 1.337 | 1.323 | 2.56|
| 1              | 10  | 1.363 | 1.263 | 0.92| 5              | 10  | 1.337 | 1.322 | 4.10|

with the standard deviation $\sigma = 1$.

In Figures 4 and 5 synthetic data sets with two input variables are used. They contain 500 and 5000 instances and are generated using the regression function:

$$s(u) = u_1 \sin(u_2) - u_2 \sin(u_2),$$

and the standard deviation $\sigma = 1$.

These figures demonstrate that the PWLREG algorithm finds the better estimation for the regression function than the SVM(Lin) algorithm and its results are comparable with those obtained by the SVM(RBF) algorithm.

5.4. Dependence on the level of noise. The synthetic data sets containing 5000 observations and one input variable are used to investigate the dependence of the performance of the PWLREG algorithm on the level of noise. These data sets are generated using the regression function (28) with the standard deviation $\sigma \in \{0, 0.5, 1\}$. Results are presented in Table 2. These results indicate that the performance of the PWLREG algorithm depends on the level of noise. The increase in the level of noise leads to the increase of values of RMSE, MAE and MSE and to the decrease of the values of CE and correlation. One can see that the RMSE and MAE measures depends rather linearly on the level of noise. Dependencies of other measures on the level of noise are close to quadratic.

5.5. Comparison of PWLREG and DGMREG algorithms. The PWLREG and DGMREG algorithms compute the same piecewise linear estimations, however, they use different optimization methods. Since they compute the similar piecewise linear functions we compare these two algorithms using the CPU time (in seconds).

Synthetic data sets with $5 \cdot 10^4$ instances and the number of input variables varying from 1 to 10 are used to demonstrate the performance of these algorithms depending on the number of input variables. Furthermore, data sets with 4 input variables and the number of instances varying from 500 to $3 \cdot 10^5$ are used to compare the performance of algorithms depending on the number of instances. Results are presented in Figure 6. These results clearly demonstrate that the PWLREG algorithm requires significantly less computational time than the DGMREG algorithm.
Figure 2. 1 input variable and 500 instances; (a) PWLREG; (b) SVM(Lin); (c) SVM(RBF).
Figure 3. 1 input variable and 5000 instances; (a) PWLREG; (b) SVM(Lin); (c) SVM(RBF).
Figure 4. (a) 2 input variables and 500 instances (a) PWLREG; (b) SVM(Lin); (c) SVM(RBF).
Figure 5. (a) 2 input variables and 5000 instances (a) PWLREG; (b) SVM(Lin); (c) SVM(RBF).
Table 2. Results for PWLREG algorithm with the different levels of noise

|        | (1,1) | (2,1) | (2,2) | (3,2) | (4,2) | (5,2) |
|--------|-------|-------|-------|-------|-------|-------|
| $\sigma=0$ |      |       |       |       |       |       |
| RMSE   | 1.094 | 1.094 | 0.063 | 0.058 | 0.055 | 0.055 |
| MAE    | 0.842 | 0.842 | 0.016 | 0.016 | 0.015 | 0.015 |
| MSE    | 1.198 | 1.198 | 0.004 | 0.003 | 0.003 | 0.003 |
| CE     | 0.013 | 0.013 | 0.997 | 0.997 | 0.998 | 0.998 |
| $r$    | 0.114 | 0.114 | 0.998 | 0.998 | 0.999 | 0.999 |
| $\sigma=0.5$ |      |       |       |       |       |       |
| RMSE   | 1.221 | 1.221 | 0.503 | 0.503 | 0.503 | 0.503 |
| MAE    | 0.937 | 0.937 | 0.404 | 0.404 | 0.404 | 0.404 |
| MSE    | 1.491 | 1.492 | 0.254 | 0.254 | 0.254 | 0.254 |
| CE     | 0.011 | 0.011 | 0.832 | 0.832 | 0.832 | 0.832 |
| $r$    | 0.105 | 0.105 | 0.912 | 0.912 | 0.912 | 0.912 |
| $\sigma=1$ |      |       |       |       |       |       |
| RMSE   | 1.498 | 1.498 | 1.242 | 1.000 | 1.000 | 1.000 |
| MAE    | 1.169 | 1.169 | 0.993 | 0.801 | 0.801 | 0.801 |
| MSE    | 2.246 | 2.247 | 1.546 | 1.003 | 1.004 | 1.005 |
| CE     | 0.008 | 0.008 | 0.318 | 0.558 | 0.558 | 0.558 |
| $r$    | 0.087 | 0.087 | 0.564 | 0.747 | 0.747 | 0.747 |

Figure 6. (a) CPU time vs the number of input variables; (b) CPU time vs the number of instances.
5.6. **Prediction performance of PWLREG algorithm on real data sets.** In this subsection, we test the proposed algorithm as a prediction method and compare it with the MLR, SVM(Lin), SVM(RBF), MARS and ANN algorithms. For this purpose, we consider five real world data sets. The brief description of these data sets is given in Table 3. Their detailed description can be found in [2].

These data sets are divided into training (80% of the whole data) and test (20% of the whole data) sets. The training data are normalized so that each input variable had the mean value 0 and the standard deviation 1. An algorithm is trained using training set and its prediction performance is checked using the test set.

| Data set                              | m    | n    |
|---------------------------------------|------|------|
| 1. Airfoil Self-noise                 | 1503 | 5    |
| 2. Red Wine Quality                   | 1599 | 11   |
| 3. White Wine Quality                 | 4898 | 11   |
| 4. Combined Cycle Power Plant         | 9568 | 4    |
| 5. Physicochemical Properties of Protein Tertiary Structure | 45730 | 9    |

Results are presented in Table 4. These results show that the PWLREG algorithm is a better predictor than the MLR and SVM(Lin) algorithms. Pairwise comparison with the SVM(RBF) algorithm shows that in two data sets the PWLREG algorithm outperforms the SVM(RBF), in two other data sets the SVM(RBF) provides better prediction results and in one data set these two algorithms give comparable results. This means that for these five data sets the PWLREG and SVM(RBF) are comparable prediction algorithms. The PWLREG algorithm outperforms the MARS in three data sets and their results are comparable in Red Wine Quality and Combined Cycle Power Plant data sets. Finally, comparison with the ANN algorithm shows that the PWLREG algorithm produces better results than the ANN in four data sets. Only in Red Wine Quality data set results by these two algorithms are comparable.

In medium size data sets (Airfoil self-noise, Red Wine Quality and White Wine Quality) with the number of data points \( m < 5000 \) the PWLREG algorithm either outperforms other algorithms or demonstrates the similar performance in comparison with the best performing algorithms (with the SVM(RBF) in the Airfoil self-noise data set and with the ANN in the White Wine Quality data set). In relatively large data sets (Combined Cycle Power Plant and Physicochemical Properties of Protein Tertiary Structure) with \( m > 5000 \) the SVM(RBF) obtains the best results and the PWLREG algorithm achieves the second best results. There is no any clear indication on the dependence of the performance of algorithms on the number of input variables.

6. **Conclusions.** In this paper, we introduced an algorithm for solving the piecewise linear regression problem. This problem is formulated as the nonsmooth non-convex optimization problem. The objective function of the optimization problem is the difference of convex functions, whose DC representation is not easy to express explicitly. However, the subdifferentials of the DC components can be formulated explicitly. Therefore, we developed an algorithm which uses information on these
subdifferentials and the objective function. We proved that this algorithm converges to critical points of the piecewise linear regression problem.

Using synthetic data sets we demonstrated that the proposed algorithm has a good generalization ability, however this ability may depend on the number of linear functions and the size of a data set. We also used synthetic data sets to compare the computational time used by the proposed algorithm and another piecewise linear regression algorithm introduced in [4]. Results demonstrate that the proposed algorithm is a significant improvement of the algorithm from [4] in the sense of used computational time.

We applied the proposed algorithm to real world data sets to test its prediction ability and to compare it with other five prediction algorithms: the MLR, SVM(Lin), SVM(RBF), MARS and ANN. The performance of these algorithms are compared using five statistical measures for prediction. Although the proposed

|                  | PWLREG | MLR  | SVM(Lin) | SVM(RBF) | MARS  | ANN   |
|------------------|--------|------|----------|----------|-------|-------|
| **Airfoil Self-noise** |        |      |          |          |       |       |
| RMSE             | 4.445  | 5.024| 5.037    | 4.710    | 5.672 | 5.499 |
| MAE              | 3.404  | 3.905| 3.877    | 3.327    | 4.614 | 4.342 |
| MSE              | 24.641 | 25.750| 25.885   | 22.941   | 32.822| 31.059|
| CE               | 0.653  | 0.558| 0.556    | 0.611    | 0.437 | 0.471 |
| r                | 0.875  | 0.761| 0.760    | 0.825    | 0.704 | 0.709 |
| **Red Wine Quality** |        |      |          |          |       |       |
| RMSE             | 0.639  | 0.643| 0.657    | 0.735    | 0.658 | 0.636 |
| MAE              | 0.489  | 0.490| 0.489    | 0.566    | 0.503 | 0.482 |
| MSE              | 0.480  | 0.430| 0.450    | 0.569    | 0.450 | 0.423 |
| CE               | 0.319  | 0.312| 0.282    | 0.101    | 0.280 | 0.326 |
| r                | 0.582  | 0.570| 0.537    | 0.394    | 0.555 | 0.585 |
| **White Wine Quality** |        |      |          |          |       |       |
| RMSE             | 0.637  | 0.690| 0.717    | 0.675    | 0.713 | 0.693 |
| MAE              | 0.508  | 0.532| 0.551    | 0.522    | 0.527 | 0.545 |
| MSE              | 0.449  | 0.482| 0.521    | 0.463    | 0.515 | 0.487 |
| CE               | 0.330  | 0.216| 0.153    | 0.249    | 0.164 | 0.209 |
| r                | 0.616  | 0.494| 0.467    | 0.521    | 0.537 | 0.520 |
| **Combined Cycle Power Plant** |        |      |          |          |       |       |
| RMSE             | 4.163  | 4.484| 4.503    | 3.907    | 4.187 | 4.283 |
| MAE              | 3.276  | 3.572| 3.559    | 2.920    | 3.295 | 3.396 |
| MSE              | 17.702 | 20.159| 20.341   | 15.377   | 17.577| 18.411|
| CE               | 0.942  | 0.933| 0.932    | 0.949    | 0.942 | 0.939 |
| r                | 0.970  | 0.966| 0.966    | 0.975    | 0.971 | 0.969 |
| **Physicochemical Properties Protein** |        |      |          |          |       |       |
| RMSE             | 4.802  | 5.203| 5.302    | 4.321    | 5.044 | 5.783 |
| MAE              | 3.840  | 4.374| 4.240    | 3.018    | 4.146 | 4.987 |
| MSE              | 23.061 | 27.101| 28.145   | 18.699   | 25.470| 33.487|
| CE               | 0.390  | 0.285| 0.257    | 0.507    | 0.328 | 0.116 |
| r                | 0.625  | 0.534| 0.528    | 0.719    | 0.573 | 0.341 |
algorithm is not always the best prediction method these results show that it is a good alternative to other mainstream prediction methods.

Acknowledgments. The authors would like to thank three anonymous referees and the handling editor for their comments that helped us to improve the quality of the paper.

REFERENCES

[1] L. T. H. An and P. D. Tao, The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems, *Annals of Operations Research*, 133 (2005), 23–46.

[2] K. Bache and M. Lichman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, [http://archive.ics.uci.edu/ml](http://archive.ics.uci.edu/ml), 2013.

[3] A. M. Bagirov, C. Clausen and M. Kohler, Estimation of a regression function by maxima of minima of linear functions, *IEEE Transactions on Information Theory*, 55 (2009), 833–845.

[4] A. M. Bagirov, C. Clausen and M. Kohler, An algorithm for the estimation of a regression function by continuous piecewise linear functions, *Computational Optimization and Applications*, 45 (2010), 159–179.

[5] A. M. Bagirov, C. Clausen and M. Kohler, An $l_2$-boosting algorithm for estimation of a regression function, *IEEE Transactions on Information Theory*, 56 (2010), 1417–1429.

[6] A. M. Bagirov, B. Karasözen and M. Sezer, Discrete gradient method: A derivative-free method for nonsmooth optimization, *Journal of Optimization Theory and Applications*, 137 (2008), 317–334.

[7] A. M. Bagirov, N. Karmitsa and M. Mäkelä, *Introduction to Nonsmooth Optimization*, Cham, Springer, 2014.

[8] A. M. Bagirov, S. Taheri and J. Ugon, Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems, *Pattern Recognition*, 53 (2016), 12–24.

[9] S. G. Bartels, L. Kuntz and S. Scholtes, Continuous selections of linear functions and nonsmooth critical point theory, *Nonlinear Analysis: Theory, Methods & Applications*, 24 (1995), 385–407.

[10] L. Breiman, J. H. Friedman, C. J. Stone and R. A. Olshen, *Classification and Regression Trees*, CRC press, 1984.

[11] R. Collobert and S. Bengio, SVMTorch: Support vector machines for large-scale regression problems, *Journal of Machine Learning Research*, 1 (2001), 143–160.

[12] V.F. Demyanov and A.M. Rubinov, *Constructive Nonsmooth Analysis*, Springer, 1995.

[13] J.H. Fridedman, Multivariate adaptive regression splines (with discussion), *Annals of Statistics*, 19 (1991), 79–141.

[14] J. Friedman, T. Hastie and R. Tibshirani, *The Elements of Statistical Learning*, Springer, Berlin, 2001.

[15] V. V. Gorokhovik, O. I. Zorko and G. Birkhoff, Piecewise affine functions and polyhedral sets, *Optimization*, 31 (1994), 209–221.

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

[17] R. Horst and N. V. Thoai, DC programming: Overview, *Journal of Optimization Theory and Applications*, 103 (1999), 1–43.

[18] K. Joki, A. M. Bagirov, N. Karmitsa and M. M. Mäkelä, A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes, *Journal of Global Optimization*, 68 (2017), 501–535.

[19] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C. C. Chang and C. C. Lin, Misc functions of the department of statistics, probability theory group, TU Wien, Package “e1071”, [http://cran.r-project.org/web/packages/e1071/e1071.pdf](http://cran.r-project.org/web/packages/e1071/e1071.pdf), 2017.

[20] S. Milborrow, Multivariate adaptive regression splines, Package “earth”, [http://cran.r-project.org/web/packages/earth/earth.pdf](http://cran.r-project.org/web/packages/earth/earth.pdf), 2017.

[21] J. Nash and J. Sutcliffe, River flow forecasting through conceptual models part I-A discussion of principles, *Journal of Hydrology*, 10 (1970), 282–290.

[22] B. Ripley and W. Venables, Feed-forward neural networks and multinomial log-linear models, Package “nnet”, [http://cran.r-project.org/web/packages/nnet/nnet.pdf](http://cran.r-project.org/web/packages/nnet/nnet.pdf), 2016.
[23] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, *Statistics and Computing*, 14 (2004), 199–222.

[24] P. D. Tao and L. T. H. An, Convex analysis approach to DC programming: Theory, algorithms and applications, *Acta Mathematica Vietnamica*, 22 (1997), 289–355.

[25] H. Tuy, *Convex Analysis and Global Optimization*, Nonconvex Optimization and Its Applications, Vol. 22. Kluwer, Dordrecht, 1998.

[26] P. H. Wolfe, Finding the nearest point in a polytope, *Mathematical Programming*, 11 (1976), 128–149.

Received August 2017; 1st revision January 2018; 2nd revision February 2018.

E-mail address: a.bagirov@federation.edu.au
E-mail address: s.taheri@federation.edu.au
E-mail address: asadi.soodabeh@gmail.com