UNIVERSITY OF TURKU

This is an Accepted Manuscript version of the article published originally in the journal:

Applied and Computational Mathematics

This version may differ from the original in pagination and typographic details. When using please cite the original.

AUTHOR(S)	Bagirov, A.M.; Taheri, S.; Karmitsa, N.; Joki, K.; Mäkelä, M.M.
TITLE	Nonsmooth DC optimization support vector machines method for piecewise linear regression
YEAR	2024
DOI	10.30546/1683-6154.23.3.2024.282
CITATION	Bagirov, A.M.; Taheri, S.; Karmitsa, N.; Joki, K.; Mäkelä, M.M. Nonsmooth DC optimization support vector machines method for piecewise linear regression (2024). *Applied and Computational Mathematics*, 23(3), 282–306. https://doi.org/10.30546/1683-6154.23.3.2024.282
VERSION	Accepted Manuscript
LICENSE	© 2024 Applied and Computational Mathematics

Appl. Comput. Math. V.23, N.3, Special Issue, 2024, pp.282-306
DOI: 10.30546/1683-6154.23.3.2024.282

NONSMOOTH DC OPTIMIZATION SUPPORT VECTOR MACHINES METHOD FOR PIECEWISE LINEAR REGRESSION

A.M. BAGIROV[1], S. TAHERI[2], N. KARMITSA[3], K. JOKI[4], M.M. MÄKELÄ[4]

Abstract. A new regression method called the adaptive piecewise linear support vector regression (A-PWLSVR) is introduced. We use the L_1-risk function to define regression errors and apply the support vector machine approach in combination with the piecewise linear regression to develop a model for regression problems. We formulate the model as an unconstrained nonconvex nonsmooth optimization problem, where the objective function is represented as a difference of two convex (DC) functions. To address the nonconvexity of the problem a novel incremental approach is proposed. This approach builds the piecewise linear estimates by applying an adaptive selection procedure for the model parameters. The approach enables us to select starting points being rough approximations of the solution. The double bundle method for nonsmooth DC optimization is applied to solve the optimization problems. The proposed A-PWLSVR method is evaluated on several synthetic and real-world data sets for regression and compared with some mainstream regression methods.

Keywords: Nonsmooth Optimization, DC Optimization, Bundle Methods, Support Vector Regression, Piecewise Linear Regression, L_1-Risk.

AMS Subject Classification: 65K05, 90C26.

1. Introduction

Support vector machine (SVM) for regression is a well-known technique in regression analysis [39,41]. Unlike many other techniques, the SVM for regression (SVR) and its modifications are less sensitive to noise [12,34,47]. The use of kernels allows the SVR to apply nonlinear functions to fit the regression function. However, the kernels are usually unknown for a given data set or may contain many parameters to be defined a priori. Several methods have been developed to optimize the kernels and to select the regression parameters. For instance, the methods proposed in [33, 38, 40] use the single kernel function while those designed in [26, 45] utilize mixed kernel functions. Furthermore, various hybrid approaches are developed to determine the parameters, for example, in [9, 11, 14, 25, 28, 29].

In this paper, we introduce a different approach for modeling and solving regression problems. In this approach, the space of continuous piecewise linear (PWL) functions, represented as a maximum of minima (or a minimum of maxima) of linear functions, are used to fit the regression function. The regression problem is formulated by applying this representation, the SVR approach, and L_1-risk. Such an approach does not require the appropriate selection of

[1]The Centre for Smart Analytics, Federation University Australia, Ballarat, Australia
e-mail: a.bagirov@federation.edu.au
[2]Mathematical Sciences, RMIT University, Melbourne, Australia
[3]Department of Computing, University of Turku, FI-20014 Turku, Finland
e-mail: napsu@karmitsa.fi
[4]Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland
e-mail: kjjoki@utu.fi, makela@utu.fi
*Corresponding author e-mail: sona.taheri@rmit.edu.au
Manuscript received 27 August 2023.

the kernels as the nonlinear fit functions can be approximated by the PWL functions. Furthermore, in contrast to the usual kernel-based SVR methods, we can use the primal form of the optimization problem and there is no necessity to formulate the dual problem. We represent the objective function in the regression problem as a difference of two convex (DC) functions. We call this regression problem the PWLSVR problem and its corresponding DC representation, the DC-PWLSVR problem.

The DC-PWLSVR problem is both nonconvex and nonsmooth. To deal with the nonconvexity of this problem, we introduce a novel adaptive incremental (A-Inc) approach: the PWL estimate is constructed incrementally starting from one affine function. This approach enables adaptive model selection as the number of minimum functions under the maximum is not given a priori and this number is determined by the algorithm itself. At each iteration of the A-Inc approach, the underlying DC-PWLSVR problem is solved by applying the double bundle (DBDC) method introduced in [22]. Theoretically, this method finds Clarke stationary points of the nonsmooth DC functions, but in practice, it is often able to find even global minima. In addition to model selection, the A-Inc approach allows us to select good starting points for the DBDC method being already rough approximations of the solution and thus leading to accurate results. The proposed adaptive PWL support vector regression (A-PWLSVR) algorithm is the combination of the A-Inc approach and the DBDC method.

We evaluate the performance of the A-PWLSVR algorithm — both as an approximation and a prediction tool — using some synthetic and real-world data sets for regression and compare it with several mainstream regression methods: multivariate adaptive regression splines [18], neural networks for regression [19], support vector regression with the radial basis function kernel [39], multiple linear regression [46], random forests [10], and piecewise linear regression [4].

The main contributions of this paper compared to the existing literature are the following:

- an optimization model for regression analysis is developed based on the combination of the linear SVR and PWL approaches;
- the objective function of this model is represented as a difference of two convex functions;
- a novel adaptive regression A-PWLSVR algorithm based on the DC representation of the PWLSVR problem as well as a new adaptive parameter selection procedure are developed;
- the performance of the proposed A-PWLSVR algorithm as an approximation and a prediction tool is validated;
- a comparative assessment of the A-PWLSVR algorithm with other state-of-the-art algorithms for regression through extensive numerical experiments is performed.

The structure of the paper is as follows. Section 2 provides some theoretical background on nonsmooth DC optimization and regression analysis. The problem statement — the PWLSVR problem — and its DC representation are given in Section 3. In Section 4, the A-PWLSVR algorithm is introduced. Numerical results are reported in Section 5 and Section 6 concludes the paper.

2. Theoretical background

We start by providing theoretical background and notations used throughout the paper.

2.1. Notations and preliminaries.

The n-dimensional Euclidean space is denoted by $\mathbb{R}^n$, the inner product by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$ for $\boldsymbol{x},\ \boldsymbol{y} \in \mathbb{R}^n$ and the associated norm by $\|\boldsymbol{x}\| = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2}$.

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

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in X.$$

The Clarke generalized directional derivative of the locally Lipschitz function f at a point $\boldsymbol{x}$ with respect to a direction $\boldsymbol{u} \in \mathbb{R}^n$ is defined as in [13]:

$$f^\circ(\boldsymbol{x}, \boldsymbol{u}) = \limsup_{\boldsymbol{y} \to \boldsymbol{x}, \alpha \downarrow 0} \frac{f(\boldsymbol{y} + \alpha \boldsymbol{u}) - f(\boldsymbol{y})}{\alpha},$$

and the Clarke subdifferential is called as the set

$$\partial f(\boldsymbol{x}) = \Big\{ \boldsymbol{\xi} \in \mathbb{R}^n : \ f^\circ(\boldsymbol{x}, \boldsymbol{u}) \geq \langle \boldsymbol{\xi}, \boldsymbol{u} \rangle \quad \forall \ \boldsymbol{u} \in \mathbb{R}^n \Big\}.$$

Each vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a subgradient. For convex functions f defined on $\mathbb{R}^n$, the set $\partial f(\boldsymbol{x})$ coincides with the classical subdifferential [5, 13].

A function $f : \mathbb{R}^n \to \mathbb{R}$ is DC if there exist convex functions $f_1, \ f_2 : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n.$$

Here, $f_1 - f_2$ is called a DC representation (decomposition) of f while f_1 and f_2 are DC components of f (for more details on DC functions, see [2, 15, 21, 42–44]). The unconstrained DC programming problem is given by:

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{1}$$

In general, we have [5]

$$\partial f(\boldsymbol{x}) \subseteq \partial f_1(\boldsymbol{x}) - \partial f_2(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where the difference of two convex sets is defined using the Minkowski difference. For a point $\boldsymbol{x}^* \in \mathbb{R}^n$ to be a local minimizer of the problem (1) it is necessary that [2]

$$\partial f_2(\boldsymbol{x}^*) \subseteq \partial f_1(\boldsymbol{x}^*).$$

Points satisfying this condition are called *inf-stationary*. This condition is not always easy to check as it requires the calculation of the whole subdifferentials of the DC components. Therefore, in most algorithms the following weaker necessary conditions are used:

$$\begin{aligned} &\mathbf{0} \in \partial f(\boldsymbol{x}^*) \quad \text{(Clarke stationarity)} \quad \text{and} \\ &\partial f_1(\boldsymbol{x}^*) \cap \partial f_2(\boldsymbol{x}^*) \neq \emptyset \quad \text{(criticality)}. \end{aligned}$$

It is known that any Clarke stationary point is also critical, however, the opposite claim is not always true [22].

2.2. **Piecewise linear regression.** In regression analysis, an $\mathbb{R}^n \times \mathbb{R}$-valued random vector $(\boldsymbol{a}, b)$ with $\mathbf{E}b^2 < \infty$ ($\mathbf{E}$ is the mathematical expectation) is considered and the dependency of b on the value of $\boldsymbol{a}$ is of interest. Let A be a given data set as follows:

$$A = \{(\boldsymbol{a}^i, b_i) \in \mathbb{R}^n \times \mathbb{R} : \ i = 1, \ldots, m\}. \tag{2}$$

Here, $\boldsymbol{a}^i \in \mathbb{R}^n$ are values of n input (explanatory) variables and $b_i \in \mathbb{R}$ are their outputs (responses). The aim of the regression analysis is to find a function $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(\boldsymbol{a})$ is a "good approximation" of b. In particular, the regression estimate can be defined by using the L_1-risk

$$\sum_{i=1}^{m} |f(\boldsymbol{a}^i) - b_i|$$

over a class $\mathcal{F}$ of measurable functions $f : \mathbb{R}^n \to \mathbb{R}$. In the papers [3,4,6–8], the authors consider $\mathcal{F}$ to be a set of continuous PWL functions as follows:

$$\Big\{ f_{\boldsymbol{xy}} : \mathbb{R}^n \to \mathbb{R} \ : \ f_{\boldsymbol{xy}}(\boldsymbol{a}) = \max_{k=1,\ldots,K} \min_{j=1,\ldots,J_k} \{\langle \boldsymbol{x}^{kj}, \boldsymbol{a} \rangle + y_{kj}\}, \boldsymbol{a} \in \mathbb{R}^n, \ \boldsymbol{x} \in \mathbb{R}^{nq}, \ \boldsymbol{y} \in \mathbb{R}^q \Big\}, \tag{3}$$

where

$$\boldsymbol{x} = (x_1^{11}, \ldots, x_n^{11}, \ldots, x_1^{KJ_K}, \ldots, x_n^{KJ_K}) \in \mathbb{R}^{nq}, \quad \text{and}$$
$$\boldsymbol{y} = (y_{11}, \ldots, \ldots, y_{KJ_K}) \in \mathbb{R}^q.$$

Here, $q = \sum_{k=1}^{K} J_k$ and $K,\ J_1, \ldots, J_K \in \mathbb{N}$ are parameters. Then, the PWL regression (PWLR) problem can be formulated as the following nonconvex nonsmooth optimization problem (see [4,6]):

$$\begin{cases} \text{minimize} & \sum_{i=1}^{m} |f_{\boldsymbol{xy}}(\boldsymbol{a}^i) - b_i| \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^{nq},\ \boldsymbol{y} \in \mathbb{R}^q. \end{cases} \tag{4}$$

2.3. **Support vector linear regression.** For a data set A, given in (2), and a margin of tolerance $\epsilon > 0$, the aim of the ϵ-SVM for linear regression (ϵ-SVLR) is to find the regression coefficients $\boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}$ in the approximating function

$$f_{\boldsymbol{x}y}(\boldsymbol{a}^i) = \langle \boldsymbol{x}, \boldsymbol{a}^i \rangle + y,$$

cf. (3) with $K = J_K = 1$) such that for each point $(\boldsymbol{a}^i, b_i) \in A$ the deviation between $f_{\boldsymbol{x}y}(\boldsymbol{a}^i)$ and b_i is at most ϵ. In addition, the "flatness" of this function is important as it reduces the complexity when there are many input variables in the data set. To obtain the flatness one should look for the small values of the components of $\boldsymbol{x}$. One way to ensure this is to minimize $\|\boldsymbol{x}\|$. Thus, the SVR can be formulated as:

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{subject to} & |f_{\boldsymbol{x}y}(\boldsymbol{a}^i) - b_i| \le \epsilon,\ i = 1, \ldots, m, \\ & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}, \end{cases} \tag{5}$$

which can be easily converted into the quadratic programming problem by doubling the number of constraints. The problem (5) is convex with a strictly convex objective function having a unique solution if at least one feasible point exists. This requires the existence of the hyperplane $f_{\boldsymbol{x}y}$ approximating all points $(\boldsymbol{a}^i, b_i) \in A$ with the precision ϵ. However, this requirement is not always possible to fulfill in practice, and therefore, the constraints are often relaxed to achieve feasibility. By applying the exact penalty function method the problem (5) can be reformulated as the following unconstrained convex nonsmooth optimization problem:

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\boldsymbol{x}\|^2 + \beta \sum_{i=1}^{m} \max\left\{0, |f_{\boldsymbol{x}y}(\boldsymbol{a}^i) - b_i| - \epsilon\right\} \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}, \end{cases} \tag{6}$$

where $\beta > 0$ is a penalty coefficient.

3. Piecewise linear support vector regression

The PWLSVR problem is formulated by generalizing the SVR problem (5) and combining it with the PWLR problem (4) as follows:

$$\begin{cases} \text{minimize} & \max\limits_{k=1,\ldots,K} \max\limits_{j=1,\ldots,J_k} \|\boldsymbol{x}^{kj}\|^2 \\ \text{subject to} & |f_{\boldsymbol{xy}}(\boldsymbol{a}^i) - b_i| \le \epsilon,\ i = 1, \ldots, m, \\ & \boldsymbol{x} \in \mathbb{R}^{nq},\ \boldsymbol{y} \in \mathbb{R}^q, \end{cases} \tag{7}$$

where $f_{\boldsymbol{xy}} \in \mathcal{F}$, $q = \sum_{k=1}^{K} J_k$, and $K,\ J_1, \ldots, J_K$ are given parameters of the PWL estimate (3). The objective function in this problem expresses the flatness of vectors of coefficients of all affine functions, and it is defined as the maximum of the squared norms of all such vectors.

Remark 1. One can also use the following functions as the flatness term:

$$\sum_{k=1}^{K} \max_{j=1,\ldots,J_k} \|\boldsymbol{x}^{kj}\|^2 \text{ and } \max_{k=1,\ldots,K} \sum_{j=1}^{J_k} \|\boldsymbol{x}^{kj}\|^2.$$

Both these functions are also convex.

Similarly to (6), the problem (7) can be reformulated as the following unconstrained optimization problem:

$$\begin{cases} \text{minimize} & \max\limits_{k=1,\ldots,K} \max\limits_{j=1,\ldots,J_k} \|\boldsymbol{x}^{kj}\|^2 + \beta \sum\limits_{i=1}^{m} \max\left\{0, |f_{\boldsymbol{xy}}(\boldsymbol{a}^i) - b_i| - \epsilon\right\} \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^{nq},\ \boldsymbol{y} \in \mathbb{R}^q. \end{cases} \tag{8}$$

As mentioned above the flatness of the approximating function is used to reduce the complexity of the problem when there are many input variables. In the SVR , usually, a large number of variables appears when kernels are applied. Since we do not apply any kernel the number of variables may not increase too much and therefore, the flatness of $\boldsymbol{x}^{kj}$s is not necessarily required. Furthermore, since the problem (8) is nonconvex piecewise quadratic, it may have many local minimizers, and therefore, the flatness condition cannot guarantee the uniqueness of a solution anymore. Thus, we consider the following special case of the model (8) where the flatness term is removed:

$$\begin{cases} \text{minimize} & F(\boldsymbol{x}, \boldsymbol{y}) = \sum\limits_{i=1}^{m} \max\left\{0, |f_{\boldsymbol{xy}}(\boldsymbol{a}^i) - b_i| - \epsilon\right\} \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^{nq},\ \boldsymbol{y} \in \mathbb{R}^q. \end{cases} \tag{9}$$

This formulation can be generalized to any PWLSVR beyond the L_1-risk function.

In general, a nonconvex PWL function like F is not necessarily subdifferentially regular [5] and thus the calculation of its subgradients is not always an easy task. However, this function is DC and subgradients of DC components can be efficiently calculated. Next, we represent the DC decomposition of the PWLSVR function F [20, 43]. First, we define

$$\psi_{ik}(\boldsymbol{x}, \boldsymbol{y}) = \max_{j=1,\ldots,J_k} \left\{ -\langle \boldsymbol{x}^{kj}, \boldsymbol{a}^i \rangle - y_{kj} \right\},\ i = 1, \ldots, m,\ k = 1, \ldots, K,$$

and

$$\varphi_i(\boldsymbol{x}, \boldsymbol{y}) = \max_{k=1,\ldots,K} \{-\psi_{ik}(\boldsymbol{x}, \boldsymbol{y})\}, \quad i = 1, \ldots, m. \tag{10}$$

For $K = 1$, we have $\varphi_i(\boldsymbol{x}, \boldsymbol{y}) = -\psi_{i1}(\boldsymbol{x}, \boldsymbol{y})$. Here, we consider two different cases: $J_1 = 1$ and $J_1 > 1$. In the first case, the function φ_i is linear and thus convex. Then, we get $\varphi_{i1}(\boldsymbol{x}, \boldsymbol{y}) = \varphi_i(\boldsymbol{x}, \boldsymbol{y})$ and $\varphi_{i2}(\boldsymbol{x}, \boldsymbol{y}) = 0,\ i = 1, \ldots, m$. In the second case, $K = 1$ and $J_1 > 1$, the function $-\varphi_i$ is convex, and thus we have $\varphi_{i1}(\boldsymbol{x}, \boldsymbol{y}) = 0$ and $\varphi_{i2}(\boldsymbol{x}, \boldsymbol{y}) = -\varphi_i(\boldsymbol{x}, \boldsymbol{y}),\ i = 1, \ldots, m$. If $K > 1$, then the function φ_i can be represented using the following DC components:

$$\varphi_{i1}(\boldsymbol{x}, \boldsymbol{y}) = \max_{k=1,\ldots,K} \Big\{ \sum_{t=1, t\neq k}^{K} \psi_{it}(\boldsymbol{x}, \boldsymbol{y}) \Big\},$$

and

$$\varphi_{i2}(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} \psi_{ik}(\boldsymbol{x},\boldsymbol{y}).$$

Let

$$\phi_i(\boldsymbol{x},\boldsymbol{y}) = |\varphi_i(\boldsymbol{x},\boldsymbol{y}) - b_i| - \epsilon, \quad i = 1,\ldots,m,$$

then, the DC components of this function are given by:

$$\phi_{i1}(\boldsymbol{x},\boldsymbol{y}) = \max\left\{2\varphi_{i1}(\boldsymbol{x},\boldsymbol{y}) - b_i, 2\varphi_{i2}(\boldsymbol{x},\boldsymbol{y}) + b_i\right\},$$

and

$$\phi_{i2}(\boldsymbol{x},\boldsymbol{y}) = \varphi_{i1}(\boldsymbol{x},\boldsymbol{y}) + \varphi_{i2}(\boldsymbol{x},\boldsymbol{y}) + \epsilon.$$

Since the maximum of a finite number of convex functions is convex, the function ϕ_{i1} is convex. Furthermore, the function φ_{i2} as a sum of convex functions is convex. Next, consider the function

$$\Psi_i(\boldsymbol{x},\boldsymbol{y}) = \max\left\{0, \phi_i(\boldsymbol{x},\boldsymbol{y})\right\}, \quad i = 1,\ldots,m,$$

which can be represented as a DC function $\Psi_i(\boldsymbol{x},\boldsymbol{y}) = \Psi_{i1}(\boldsymbol{x},\boldsymbol{y}) - \Psi_{i2}(\boldsymbol{x},\boldsymbol{y})$ with

$$\Psi_{i1}(\boldsymbol{x},\boldsymbol{y}) = \max\left\{\phi_{i1}(\boldsymbol{x},\boldsymbol{y}), \phi_{i2}(\boldsymbol{x},\boldsymbol{y})\right\},$$

and

$$\Psi_{i2}(\boldsymbol{x},\boldsymbol{y}) = \phi_{i2}(\boldsymbol{x},\boldsymbol{y}).$$

Then, the function F can be rewritten as:

$$F(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \Psi_i(\boldsymbol{x},\boldsymbol{y}),$$

and its DC representation is $F(\boldsymbol{x},\boldsymbol{y}) = F_1(\boldsymbol{x},\boldsymbol{y}) - F_2(\boldsymbol{x},\boldsymbol{y})$ with the DC components

$$F_1(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \Psi_{i1}(\boldsymbol{x},\boldsymbol{y}), \tag{11}$$

and

$$F_2(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \Psi_{i2}(\boldsymbol{x},\boldsymbol{y}).$$

The DC representation of the objective function in (8) can be obtained from (11) by adding the flatness term to the function F_1.

Note that DC decompositions are not unique [20, 22, 30] and the representation (11) is only one of them.

Remark 2. It is known that continuous piecewise linear functions can also be represented as a minimum of maxima of affine functions [20]. In this case, $\mathcal{F}$ is a class of functions represented as a minmax of affine functions, that is

$$\bar{\varphi}_i(\boldsymbol{x},\boldsymbol{y}) = \min_{k=1,\ldots,K} \max_{j=1,\ldots,J_k} \left\{\langle \boldsymbol{x}^{kj}, \boldsymbol{a}^i\rangle + y_{kj}\right\}, \ i = 1,\ldots,m,$$

The function $\bar{\varphi}_i$ is DC: $\bar{\varphi}_i(\boldsymbol{x},\boldsymbol{y}) = \bar{\varphi}_{i1}(\boldsymbol{x},\boldsymbol{y}) - \bar{\varphi}_{i2}(\boldsymbol{x},\boldsymbol{y})$ where

$$\bar{\varphi}_{i1}(\boldsymbol{x},\boldsymbol{y}) = \sum_{k=1}^{K} \max_{j=1,\ldots,J_k} \left\{\langle \boldsymbol{x}^{kj}, \boldsymbol{a}^i\rangle + y_{kj}\right\},$$

and

$$\bar{\varphi}_{i2}(\boldsymbol{x},\boldsymbol{y}) = \max_{k=1,\ldots,K} \sum_{t=1,t\neq k}^{K} \max_{j=1,\ldots,J_t} \left\{\langle \boldsymbol{x}^{tj}, \boldsymbol{a}^i\rangle + y_{tj}\right\}.$$

Using these functions instead of φ_i, defined in (10), leads to a model with a similar complexity of the DC components F_i given in (11) than before. Therefore, there is no significant difference in between these two DC representations.

4. Adaptive piecewise linear support vector regression algorithm

To solve the DC-PWLSVR problem (9), we introduce the A-PWLSVR algorithm. First, we present this algorithm in Fig.1. Then, we describe its details and give its step-by-step form in Algorithm 1.

In the A-PWLSVR algorithm, the PWL estimate $f_{\boldsymbol{xy}}$ is built by applying the new A-Inc approach. This approach finds adaptively a constructive final estimate and consists of two loops. In the first loop, we start with only one minimum function under the maximum and calculate linear functions under the minimum. Once the linear functions are calculated we move on to the second loop, where more minimum functions under the maximum are adaptively added. The A-Inc stops if the current estimate is good enough or the required number of minimum functions is reached.

Generally, the number $J_k,\ k = 1, \ldots, K$ of linear functions under a minimum in the problem (9) may vary for different values of k. However, in what follows, we assume that $J_k \equiv J$ is a constant. Although such an assumption narrows the space over which we can find the PWL estimates of the regression function, it considerably reduces the computational time required by the algorithm. The numbers K and J defining the PWL estimate are user-defined parameters and need to be given a priori. Note that K is an upper limit for the number of minimum functions, and the exact value of $k \leq K$ is chosen by the A-PWLSVR algorithm. That is the algorithm stops with $k < K$ if the current model is already good enough and adding more minimum functions does not make any sufficient decrease in the objective function values between iterations. Since the PWL estimate $f_{\boldsymbol{xy}}$ is built gradually, we use the notation (k, j)-DC-PWLSVR in the A-PWLSVR algorithm. This shows the DC-PWLSVR problem with k minimum functions under the maximum where each minimum function consists of j linear functions.

In addition to model selection, the A-Inc approach enables us to efficiently generate good starting points for the nonconvex DC-PWLSVR problem. This is an important aspect as this problem is, primarily, nonconvex and we are solving it with a local search method DBDC [22]. The DBDC method is originally developed for solving general nonsmooth DC optimization problems. It utilizes explicitly the DC representation of the objective function to take advantage of both the convexity and the concavity of the objective. In other words, the convex cutting plane models $\hat{F}_1$ and $\hat{F}_2$ (see, e.g. [23]) are formed for both DC components F_1 and F_2 of the objective $F = F_1 - F_2$ and combined to get the nonconvex DC model $\hat{F} = \hat{F}_1 - \hat{F}_2$. This nonconvex cutting plane model represents the nonconvex objective function better than the classical convex one. To build the model, the DBDC method collects subgradient information from the previous iterations into bundles to approximate the subdifferentials of the DC components. The bundles are used to generate a better model of the objective function if the descent condition, in other words, the serious step, is not satisfied. In addition, the DBDC method uses the escape procedure [22] to guide the algorithm to pass critical points which are not Clarke stationary. Therefore, the DBDC method ends up with Clarke stationary points of the DC-PWLSVR problem. Algorithm 1 is the step-by-step form of the A-PWLSVR algorithm.

Remark 3. The A-PWLSVR algorithm applies the DBDC method to solve underlying DC optimization problems. The DBDC method is the version of the bundle method for solving DC optimization problems. It is well-known that bundle methods have, in general, at most the linear rate of convergence [5]. Since we consider only a finite number of linear functions under the minimum and the finite number of minimum functions under the maximum, the A-PWLSVR algorithm also has a linear rate of convergence.

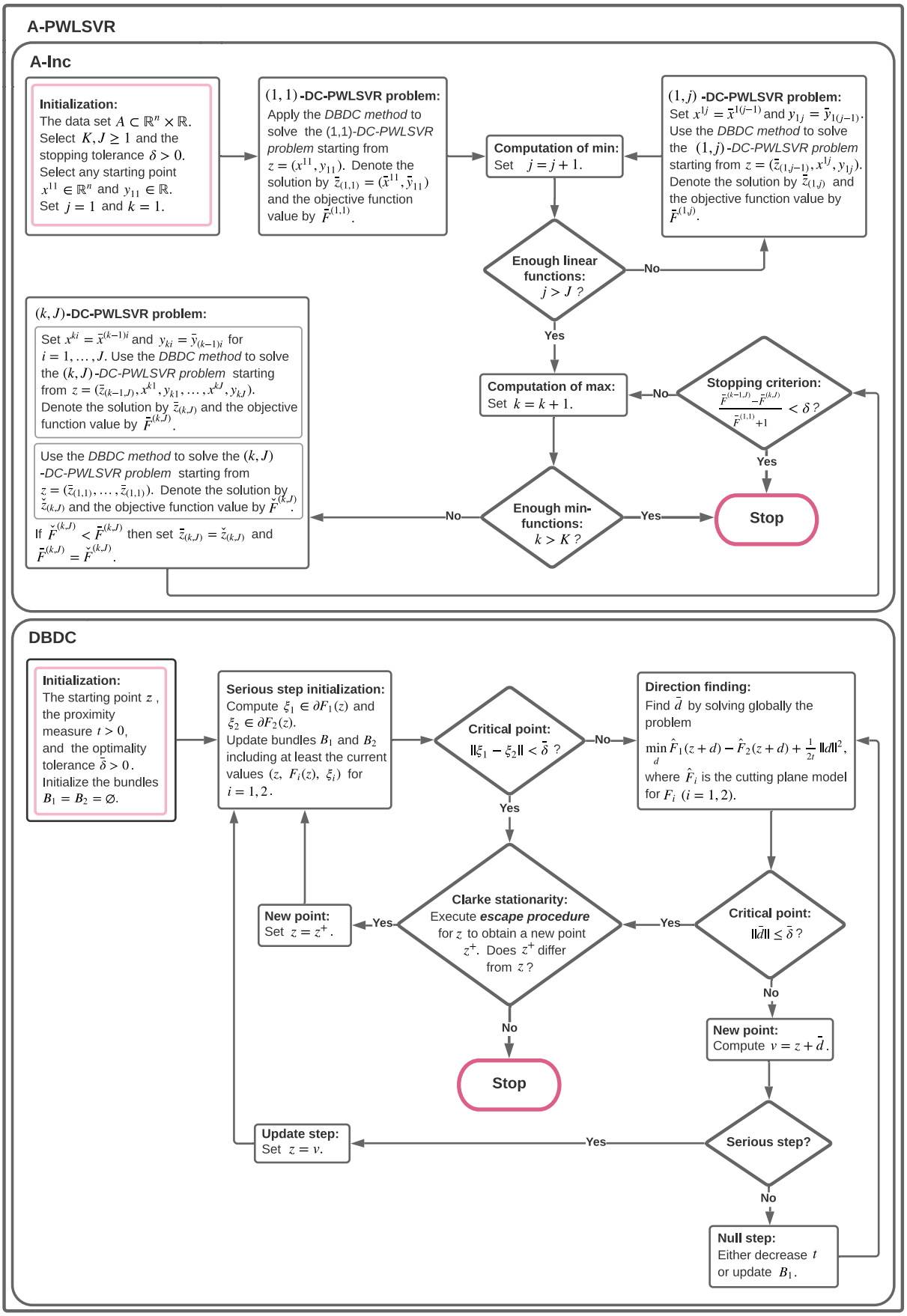


FIGURE 1. A-PWLSVR algorithm. In the DBDC method, $\boldsymbol{\xi}_i \in \partial F_i(\boldsymbol{z})$ $(i = 1, 2)$ are arbitrary elements (subgradients) from the subdifferentials.

Algorithm 1 A-PWLSVR algorithm

Require: The data set $A \subset \mathbb{R}^n \times \mathbb{R}$, the stopping tolerance $\delta > 0$ and the numbers $K,\ J \geq 1$ for computing maximum and minimum functions in the PWL function, respectively.

Ensure: The PWL function represented as a maximum of minima of linear functions approximating the data set A.

1: [*Initialization*] Select any starting point $\boldsymbol{x}^{11} \in \mathbb{R}^n$ and $y_{11} \in \mathbb{R}$.

2: [(1,1)-DC-PWLSVR *problem*] Apply the DBDC method to solve the $(1,1)$-DC-PWLSVR problem (9) starting from $\boldsymbol{z} = (\boldsymbol{x}^{11}, y_{11}) \in \mathbb{R}^{n+1}$. Denote the solution by $\bar{\boldsymbol{z}}_{(1,1)} = (\bar{\boldsymbol{x}}^{11}, \bar{y}_{11})$ and the objective function value by $\bar{F}^{(1,1)}$. Set $k = 1$ and $j = 1$.

3: [*Computation of* min] Set $j = j + 1$.

if $j > J$ **then**
go to Step 5.
end if

4: [(1,j)-DC-PWLSVR *problem*] Set $\boldsymbol{x}^{1j} = \bar{\boldsymbol{x}}^{1(j-1)}$ and $y_{1j} = \bar{y}_{1(j-1)}$. Apply the DBDC method to solve the $(1,j)$-DC-PWLSVR problem (9) starting from

$$\boldsymbol{z} = (\bar{\boldsymbol{z}}_{(1,j-1)}, \boldsymbol{x}^{1j}, y_{1j}) \in \mathbb{R}^{(n+1)j}.$$

Denote the solution by

$$\bar{\boldsymbol{z}}_{(1,j)} = (\bar{\boldsymbol{x}}^{11}, \bar{y}_{11}, \ldots, \bar{\boldsymbol{x}}^{1(j-1)}, \bar{y}_{1(j-1)}, \bar{\boldsymbol{x}}^{1j}, \bar{y}_{1j})$$

and the objective function value by $\bar{F}^{(1,j)}$. Go to Step 3.

5: [Computation of max] Set $k = k + 1$.

6: [Stopping criterion]

if $k > K$ **then**
stop
end if

7: [(k,J)-DC-PWLSVR *problem*]
(i) Set $\boldsymbol{x}^{ki} = \bar{\boldsymbol{x}}^{(k-1)i}$ and $y_{ki} = \bar{y}_{(k-1)i}$ for $i = 1, \ldots, J$. Apply the DBDC method to solve the (k,J)-DC-PWLSVR problem (9) starting from

$$\boldsymbol{z} = (\bar{\boldsymbol{z}}_{(k-1,J)}, \boldsymbol{x}^{k1}, y_{k1}, \ldots, \boldsymbol{x}^{kJ}, y_{kJ}) \in \mathbb{R}^{k(n+1)J}.$$

Denote the solution by

$$\bar{\boldsymbol{z}}_{(k,J)} = (\bar{\boldsymbol{x}}^{11}, \bar{y}_{11}, \ldots, \bar{\boldsymbol{x}}^{1J}, \bar{y}_{1J}, \ldots, \bar{\boldsymbol{x}}^{k1}, \bar{y}_{k1}, \ldots, \bar{\boldsymbol{x}}^{kJ}, \bar{y}_{kJ})$$

and the objective function value by $\bar{F}^{(k,J)}$.

(ii) Apply the DBDC method to solve the (k,J)-DC-PWLSVR problem (9) starting from

$$\boldsymbol{z} = (\bar{\boldsymbol{z}}_{(1,1)}, \ldots, \bar{\boldsymbol{z}}_{(1,1)}) \in \mathbb{R}^{k(n+1)J}.$$

Denote the solution by

$$\check{\boldsymbol{z}}_{(k,J)} = (\check{\boldsymbol{x}}^{11}, \check{y}_{11}, \ldots, \check{\boldsymbol{x}}^{1J}, \check{y}_{1J}, \ldots, \check{\boldsymbol{x}}^{k1}, \check{y}_{k1}, \ldots, \check{\boldsymbol{x}}^{kJ}, \check{y}_{kJ})$$

and the objective function value by $\check{F}^{(k,J)}$.

if $\check{F}^{(k,J)} < \bar{F}^{(k,J)}$ **then**
set $\bar{\boldsymbol{z}}_{(k,J)} = \check{\boldsymbol{z}}_{(k,J)}$ and $\bar{F}^{(k,J)} = \check{F}^{(k,J)}$.
end if

8: [Stopping criterion]

if $(\bar{F}^{(k-1,J)} - \bar{F}^{(k,J)})/(\bar{F}^{(1,1)} + 1) < \delta$ **then**
stop
else
go to Step 5.
end if

Remark 4. Note that in Step 2 of the A-PWLSVR algorithm, one needs to solve the convex problem since the objective function is convex for $k = j = 1$. The $(1,1)$-DC-PWLSVR problem is, in fact, the SVR problem (6) without the quadratic flatness term.

Remark 5. In Step 7 of the A-PWLSVR algorithm, two different approaches are used to find the starting point to solve the (k,J)-DC-PWLSVR problem (9). In the first approach, we use

points obtained in the previous iteration of the A-Inc approach while in the second approach, the solution for the problem with one linear function is used to define the starting point.

Remark 6. The A-Inc approach enables us to find the proper PWL model for each data set. Although this approach may require more computational efforts than if one utilizes a predefined, possibly inaccurate, PWL model with a random starting point, its usage leads to a better (data specific) model and thus a better solution to the regression problem. Using many starting points in the predefined PWL model may improve the quality of the solution if the model itself is complex enough. However, this significantly increases the computational burden since if the predefined PWL model does not have enough pieces, it will not approximate data well regardless of the starting points.

5. Numerical experiments

To evaluate the performance of the proposed A-PWLSVR algorithm and to compare it with some well-known machine learning regression algorithms, we carry out numerical experiments using several synthetic and real-world data sets. Computational experiments are carried out on a laptop with Intel(R) Core(TM) i5-8250U 1.60-GHz CPU and 8-RAM GB. The A-PWLSVR algorithm is implemented in Fortran 95 and compiled using `gfortran`, the GNU Fortran compiler. The source code of this algorithm is available at `http://napsu.karmitsa.fi/a-pwlsvr` and also at GitHub: `https://github.com/SnTa2019/Regression-via-Nonsmooth-Optimization`.

The parameters of the A-PWLSVR algorithm are selected as follows:

- the stopping tolerance is set to $\delta = 0.05$. The smaller values of this parameter may lead to overfitting and the larger values may result in finding the PWL functions with only very few linear pieces not providing a good approximation of the complex regression function;
- the maximum number of minimum functions is set to $K = 5$. This number is selected to be big enough during our preliminary numerical tests, and usually, the algorithm stops before this upper limit is reached. The proper number of maximum functions $k \leq K$ is chosen based on the changes in the function values in two successive iterations. Note that since these values are computed only from the training set there is no need to use a validation set;
- the number of affine functions under minimum functions is set to $J = 3$. If less than this number is needed, then the algorithm defines a linear piece that is not used. On the other hand, the larger values of J would only increase the computational burden without any significant improvement in the solution;
- the margin of tolerance ϵ is chosen from the segment $[0.01, 0.05]$.

In the DBDC method, we use the default parameters that are given in [22].

5.1. Performance measures and data sets.

In this subsection, we describe the performance measures, used to evaluate and compare the algorithms, followed by the brief description of data sets.

5.1.1. *Performance measures.* Let $b_1, \ldots, b_m$, $m \geq 1$ be actual observed values (outputs) and $\hat{b}_1, \ldots, \hat{b}_m$ be their forecasted values. We use the following performance measures:

- *root mean square error*:

$$\text{RMSE} = \Big(\frac{1}{m}\sum_{i=1}^{m}(\hat{b}_i - b_i)^2\Big)^{1/2};$$

- *mean absolute error*:

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|\hat{b}_i - b_i|;$$

- *mean absolute percentage error*:

$$\mathrm{MAPE} = \frac{1}{m}\sum_{i=1}^{m}\Big|\frac{\hat{b}_i - b_i}{b_i}\Big|;$$

- *coefficient of determination*:

$$R^2 = 1 - \Big(\frac{\sum\limits_{i=1}^{m}(b_i - \hat{b}_i)^2}{\sum\limits_{i=1}^{m}(b_i - b_0)^2}\Big);$$

- *Pearson's correlation coefficient*:

$$r = \frac{\sum\limits_{i=1}^{m}(b_i - b_0)(\hat{b}_i - \hat{b}_0)}{\Big(\sum\limits_{i=1}^{m}(b_i - b_0)^2 \sum\limits_{i=1}^{m}(\hat{b}_i - \hat{b}_0)^2\Big)^{1/2}}.$$

Here, b_0 is the mean of observed values and $\hat{b}_0$ is the mean of predicted values. The small values of the RMSE, MAE, and MAPE measures indicate small deviations of the predictions from actual observations. Note that in the measure MAPE, we replaced $|(\hat{b}_i - b_i)/b_i|$ by $|\hat{b}_i - b_i|$ if $|b_i|$ is very small. The R^2 measure ranges from $-\infty$ to 1, where $R^2 = 1$ means a perfect prediction, $R^2 = 0$ indicates that the model predictions are as accurate as the mean of the observed data, and $-\infty < R^2 < 0$ occurs when the observed mean is a better predictor than the model. The range of r is from -1 to 1, where $r = 1$ implies that a linear equation describes the relationship between observed and predicted values perfectly, $r = -1$ means that all the data points lie on a line for which a predicted value decreases as an observed value increases and $r = 0$ happens when there is no linear relationship between the actual and observed values.

5.1.2. *Data sets.* We use both synthetic and real-world data sets in our numerical experiments. Synthetic data: The following different types of synthetic data sets are used:

(i) simple synthetic data sets with only one input variable created using
   a. linear functions with various levels of randomly generated noise and outliers;
   b. the following sinusoidal function with various levels of randomly generated noise and outliers:

$$s(u) = 2|u|\sin\Big(\frac{\pi u}{2}\Big),\ u \in \mathbb{R}.$$

   These data sets are illustrated as blue dots in Figs. 2 and 5 – 8;

(ii) the data sets that are generated using the regression function represented as the composition of the sinusoidal function

$$s(\boldsymbol{v}) = \sum_{j=1}^{n}(-1)^{j-1}v_j\sin(v_j^2),$$

   with noise, where $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$;

(iii) the data sets that are created using the piecewise linear regression function

$$s(w) = 2\max\big\{1, \min\{2w+3, -8w+3\}\big\},$$

   with noise, where $w = \sum\limits_{j=1}^{n} w_j,\ (w_1, \ldots, w_n) \in \mathbb{R}^n$.

See [7, 8] for more details of data sets (ii) and (iii).

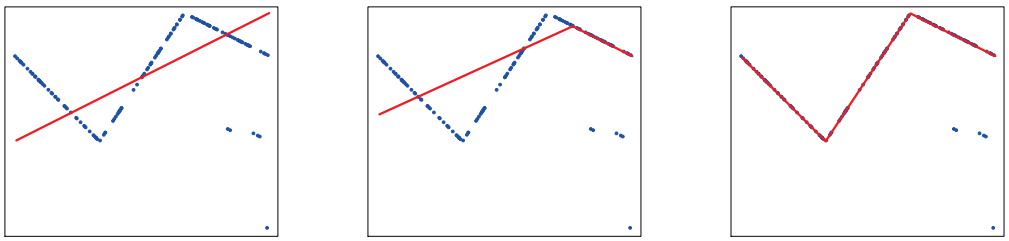
TABLE 1. Brief description of real-world data sets.

Data set	m	n
Residential building	372	107
Boston housing	506	13
Concrete compressive strength	1 030	9
Airfoil self-noise	1 503	5
Red wine quality	1 599	11
White wine quality	4 898	11
Combined cycle power plant	9 568	4
Online news popularity	39 644	58
Physicochemical properties of protein tertiary structure	45 730	9
BlogFeedback	60 021	280
SGEMM GPU kernel performance	241 600	14

Real-world data: Real-world data sets are selected from [16]. They have a varying number of data points and input variables: The number of points ranges from 372 to 241 600 and the number of input variables ranges from 4 to 280. Such a choice of data sets enables us to get a good overall view of the performance of algorithms. The data sets have only numeric attributes and no missing values. The brief description of these data sets is given in Table 1, and their detailed description can be found in [16]. Note that for Residential building and SGEMM GPU kernel performance data sets, we report results only with the first output feature as results with other output features are very similar.

5.2. **Performance of the A-PWLSVR algorithm.** In this subsection, we study the performance of the A-PWLSVR algorithm. First, we present the iterative model construction of the A-PWLSVR algorithm. Then, we demonstrate its generalization ability. Further, we analyze the behavior of the algorithm depending on the number of input variables, the number of data points, and the parameter ϵ. Finally, we provide the final models built by the A-PWLSVR algorithm in the real-world data sets and the CPU time required to construct these models.

5.2.1. *Model construction.* Fig.2 illustrates how the A-PWLSVR algorithm constructs the regression function by adding one linear function at each iteration. We use the simple synthetic data given in (i)-a for this purpose. In this figure, blue dots represent the actual data and red lines represent the estimation functions.



(A) One linear function (B) Two linear functions (C) Final model

FIGURE 2. Illustration of the A-PWLSVR algorithm at different iterations using synthetic data given in (i)-a.

5.2.2. *Generalization ability.* Using the synthetic data sets given in (ii), we study the generalization ability of the A-PWLSVR algorithm. We take the data sets containing 5 000 points with the number of input variables ranging from 1 to 10. Data sets are divided into two sets: Training (80% of all data) and test (20%) sets. The training sets are normalized such that each input variable has the mean value 0 and the standard deviation $\sigma = 1$. Results are reported in Fig.3, where we present the RMSE values for both training and test sets. These results show the good generalization property of the A-PWLSVR algorithm as the RMSE values for training and test sets are very close to each other for a different number of input variables.

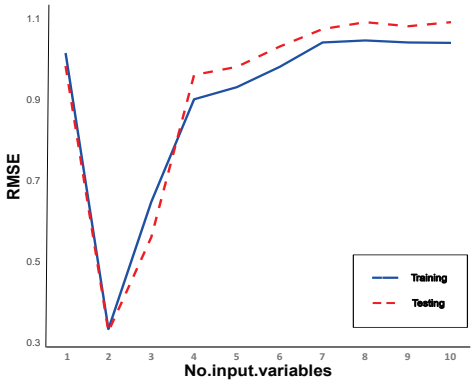


FIGURE 3. Generalization ability of the A-PWLSVR algorithm (RMSE values) using synthetic data given in (ii).

5.2.3. *Dependency on the number of input variables and the number of data points.* To study the dependence of the performance of the proposed algorithm on the number of input variables (n) and the number of data points (m), we utilize the following two versions of synthetic data sets given in (iii):

- the data set with 10 000 data points and the number of input variables ranging from 1 to 10;
- the data set with 5 input variables and the number of data points ranging from 500 to 50 000.

We apply the A-PWLSVR algorithm to both versions of data. The dependency of the CPU time (in seconds) on the number of input variables and on the number of data points is presented in Fig.4. We can see that the dependency on the number of input variables is close to quadratic. The final models obtained by the algorithm in all cases are $(3, 3)$. The dependency of the CPU time on the number of data points is roughly linear. The final models obtained with the A-PWLSVR algorithm in all cases are (2,3).

5.2.4. *Dependency on the parameter ϵ.* Since the choice of the parameter ϵ in the SVM approach is important (see, e.g. [24]) we analyze the performance of the A-PWLSVR algorithm with different values of ϵ. Here, we report the results with Airfoil self-noise data set which are presented in Table 2. It can be observed that when data is scaled properly, ϵ can be chosen, for instance, from the segment $[0.01, 0.05]$ and the difference between results using any two values of ϵ from this segment is negligible. The best result with this data set is obtained with $\epsilon = 0.01$.

5.2.5. *Final models built by the algorithm and the required CPU time.* As mentioned before, the best PWL model obtained with the A-PWLSVR algorithm is data dependent. In Table 3, we provide for each real-world data set, the final model built by the algorithm as well as the computational time (in seconds) needed to construct this model. Recall that (k, j) denotes the k minimum functions under the maximum where each minimum function consists of j linear functions.

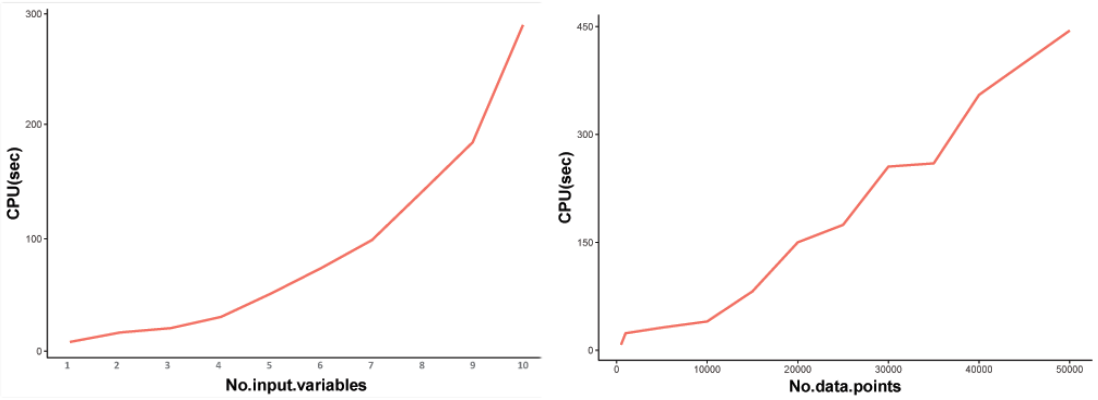


FIGURE 4. CPU time required by the A-PWLSVR algorithm using synthetic data given in (iii) with different number of input variables and data points.

TABLE 2. Results for dependence on ϵ using Airfoil self-noise data set.

ϵ	0.00	0.01	0.05	0.10	0.50	1.00
RMSE	4.009	3.892	3.991	4.364	4.138	5.323
MAE	3.025	2.884	2.897	3.296	3.164	4.013
MAPE	0.024	0.023	0.033	0.026	0.025	0.031
R^2	0.719	0.735	0.805	0.666	0.700	0.504
r	0.910	0.883	0.889	0.896	0.860	0.804

TABLE 3. Final models and required CPU times (in sec.) by the A-PWLSVR algorithm on real-world data sets.

	Residential	Boston	Concrete	Airfoil	Redwine	Whitewine	Combined	Online	Protein	BlogFeedback	SGEMM
Best model	(4,3)	(3,3)	(3,3)	(5,3)	(4,3)	(3,3)	(4,3)	(2,3)	(5,3)	(3,3)	(4,3)
CPU time	14.231	19.403	33.044	29.565	63.434	84.709	110.384	785.234	506.390	967.920	896.451

5.3. **Comparison of the the A-PWLSVR algorithm with other regression algorithms.** In this subsection, we compare the performance of the proposed algorithm with some other well-known regression algorithms.

5.3.1. *Algorithms for comparison.* We use the following algorithms in our comparison:

- Piecewise Linear Regression (PWLREG) [4];
- Random Forest Regression (RFR) [10];
- Multiple Linear Regression (MLR) [46];
- Neural Networks for Regression (NNR) [1];
- Multivariate Adaptive Regression Splines (MARS) [17];
- Support Vector Regression with the Radial Basis Function (SVR) [41].

The PWLREG algorithm is implemented in Fortran 95 and compiled using free compiler `gfortan`. For this algorithm, we use the same parameters as in the paper [4]. For other machine learning regression algorithms, we use their R implementations available in [27, 31, 32, 35], and the parameters are chosen similarly to the ones that are suggested in these references.

The PWLREG algorithm considers the problem of finding a continuous PWL function approximating a regression function using a predefined model. The objective in the regression problem is a DC function, and an algorithm is designed based on the subgradients of the DC components to find PWL functions.

The RFR algorithm is an ensemble technique that applies multiple decision trees and the Bootstrap Aggregation technique. It combines the qualities and features of multiple decision trees as base learning models to determine the final output. Row sampling and feature sampling from data forming sample data sets for every model are randomly performed. This process is called as Bootstrap.

MLR is used to determine a linear relationship between multiple independent variables and one dependent variable. Once each of the independent factors has been determined to predict the dependent variable, the information on the multiple variables can be used to create an accurate prediction on the level of effect they have on the dependent variable.

NNR consists of a system of interconnected artificial neurons (nodes) made up of three groups: Layers of "input" units, "hidden" units, and "output" units. The dimension of input variables determines the size of the input layer while the number and size of the hidden layers can be chosen more freely. The output layer consists of only one node in regression problems. For a given node, the inputs are multiplied by the weights associated with the node and summed together (summed activation of the node). The summed activation is then transformed via an activation function to define the specific output of the node.

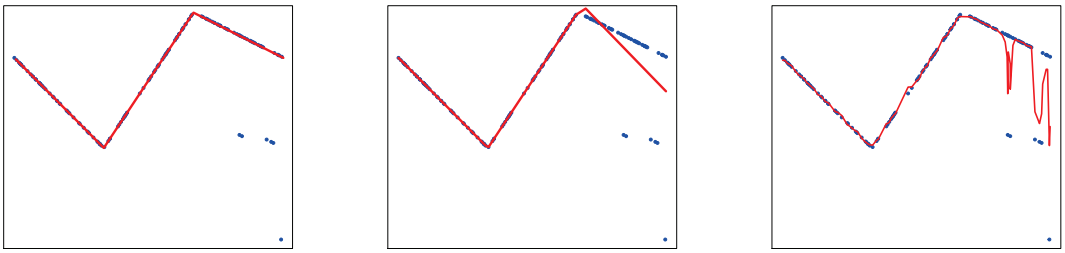
The MARS is a nonparametric method that builds multiple linear regression models across the range of input values. It does this by partitioning the data and running a linear regression model on each different partition. The MARS algorithm builds a model in two steps. First, it creates a collection of basic functions to partition the range of input data into several groups. For each group, a separate linear regression is modeled, each with its slope. The connections between the separate regression lines are automatically found by the algorithm.

The SVR is an algorithm that allows us to choose how tolerant we are of errors, both through an acceptable error margin and through tuning the tolerance of falling outside that acceptable error rate. It gives the flexibility of defining how much error is acceptable in the model and finds an appropriate hyperplane to fit the data. The aim of the SVR is to optimize the coefficients while the error term is handled in the constraints, where the absolute error is set to be less than or equal to a specified margin.

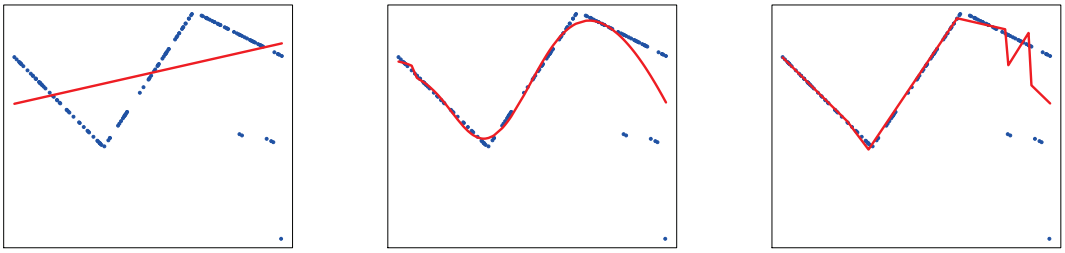
5.3.2. *The effect of outliers.* Using synthetic data sets given in (i) — with only one input variable — we present the effects of outliers to the approximations obtained by the A-PWLSVR algorithm and other algorithms given above. The results are illustrated in Figs.5–8, where blue dots are the data points and red lines are the approximations obtained by the algorithms.

In the data set, illustrated in Fig.5, A-PWLSVR and SVR are the only algorithms whose results are not strongly affected by the outliers. More accurate results by all algorithms are obtained in the data set given in Fig.6 where A-PWLSVR algorithm provides the best approximation. All algorithms, except RFR, exhibit a robust behavior in the data set illustrated in Fig.7 where NNR achieves the best approximation and MLR fails to approximate the data accurately. The data set given in Fig.8 has a clusterwise structure. In this data set, the A-PWLSVR and MARS algorithms obtain the most accurate approximations.

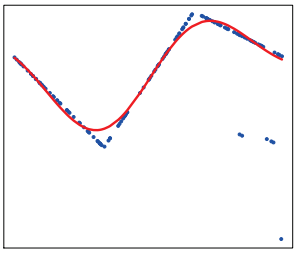
Based on the obtained results, we can conclude that the A-PWLSVR algorithm is able to approximate data with various structures and it is robust to outliers in these data sets. The SVR is also robust to outliers, nevertheless, its accuracy in approximating data with piecewise linear structure is not satisfactory. The reason for this could be the fact that this algorithm is designed to approximate smooth functions and thus it cannot fit well the data with switchings/jumpy points. The results of the PWLREG, MARS, NNR and RFR algorithms are affected by outliers and they demonstrate varying degrees of robustness. The MARS achieves high accuracy in approximating data with clusterwise structure, while the NNR approximates smooth data accurately, however, the latter algorithm fails in data sets with piecewise linear (jumpy points) and clusterwise structures. The MLR fails to approximate data in all synthetic data sets considered in this paper.



(A) A-PWLSVR (B) PWLREG (C) RFR



(D) MLR (E) NNR (F) MARS

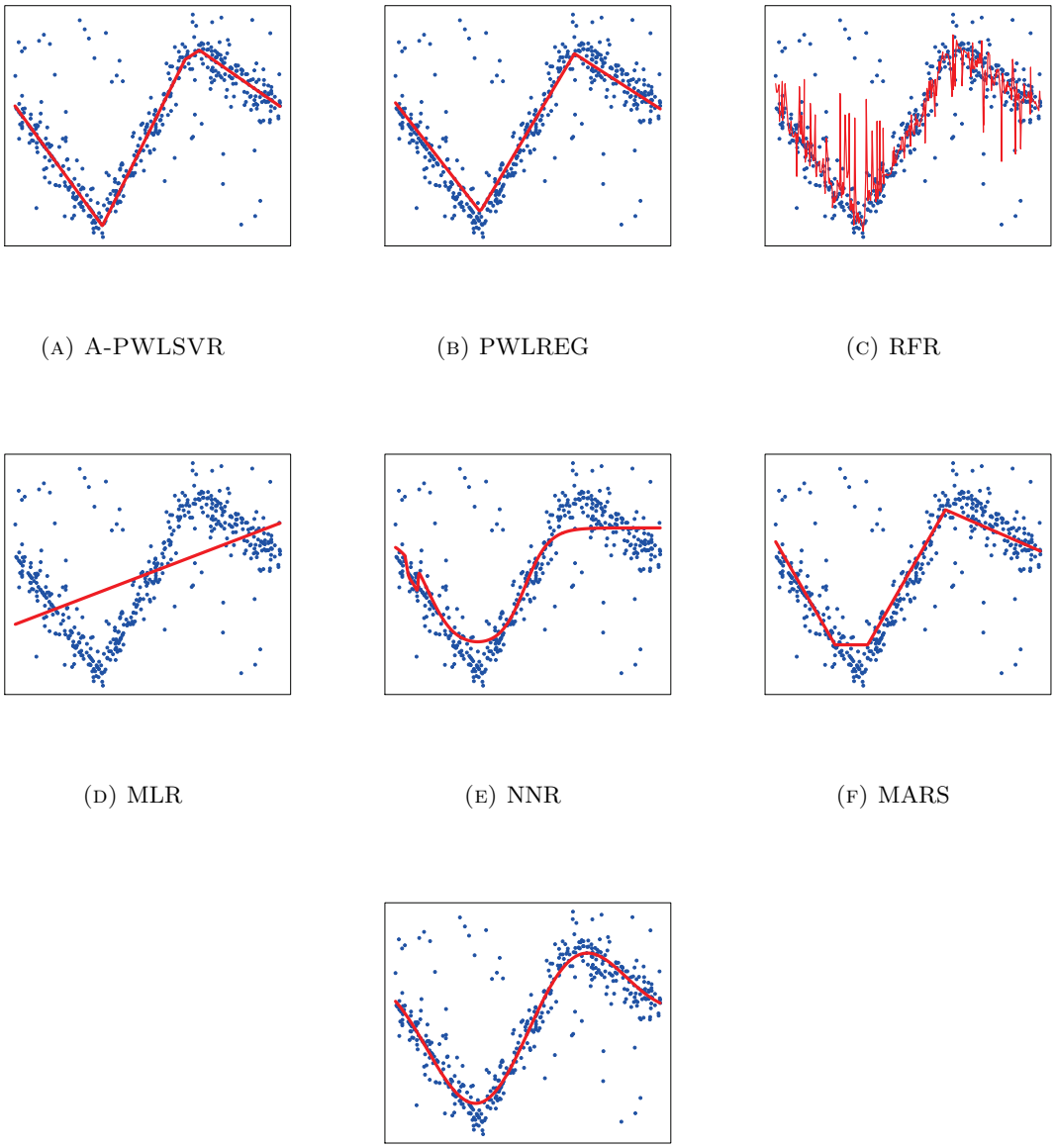


(G) SVR

FIGURE 5. Results for synthetic data given in (i)-a with small number of outliers.

5.3.3. *Results on the prediction performances of algorithms.* We study the prediction performances of the algorithms using the evaluation criteria given in Subsection 5.1. We use both synthetic and real-world data sets for this purpose. All the algorithms are trained using the training set and their prediction performances are evaluated using the test set.

Results for the synthetic data sets, generated using the regression function given in (ii) with 5 000 data points and $2, 3, 4, 7$ and 10 input variables, are presented in Fig.9. The final models obtained by the A-PWLSVR algorithm are $(5, 3)$ for $n = 2, 3, 4$ and $(3, 3)$ for $n = 7, 10$. We can see that the A-PWLSVR algorithm outperforms other algorithms by obtaining the smallest values for the RMSE and MAE when $n = 7, 10$.



(A) A-PWLSVR (B) PWLREG (C) RFR

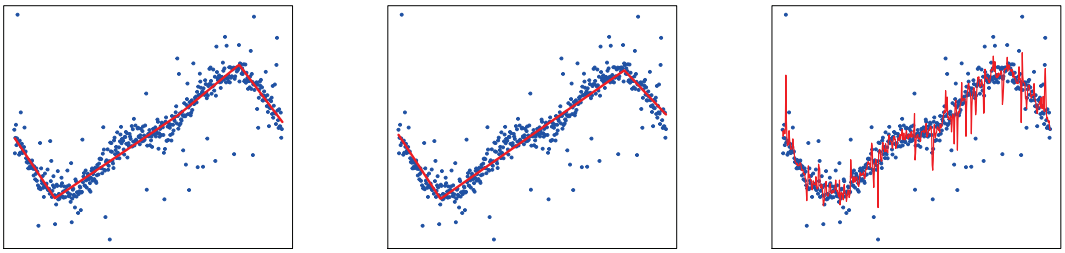
(D) MLR (E) NNR (F) MARS

(G) SVR

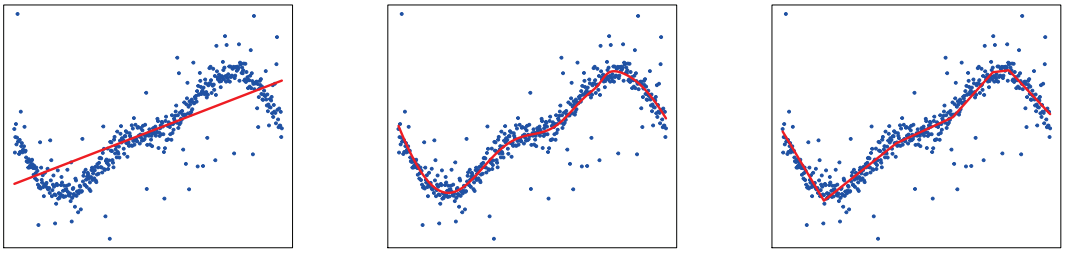
FIGURE 6. Results for synthetic data given in (i)-a with large number of outliers.

This algorithm shows a satisfactory performance in all other cases. To further analysis of the significance of these results, we applied the statistical "t-test". The statistical significance test on four performance measures is reported in Table 5, where the MLR is considered as a baseline algorithm. We see that the A-PWLSVR algorithm performs best concerning RMSE, MAE, R^2 and its improvement over the baseline algorithm is statistically significant (p-value less than < 0.05 under paired t-test). The RFR achieves the highest value of r with the p-value equal to 0.013.

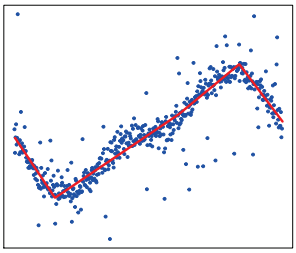
Next, we discuss and compare the results of algorithms using real-world data sets (Table 5). We apply the 5-fold cross-validation.



(A) A-PWLSVR (B) PWLREG (C) RFR



(D) MLR (E) NNR (F) MARS

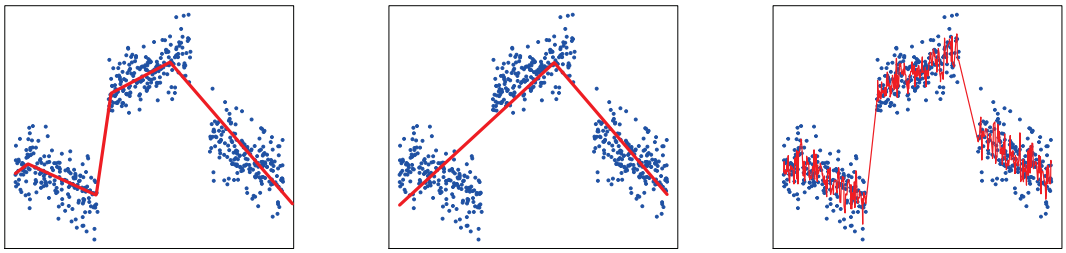


(G) SVR

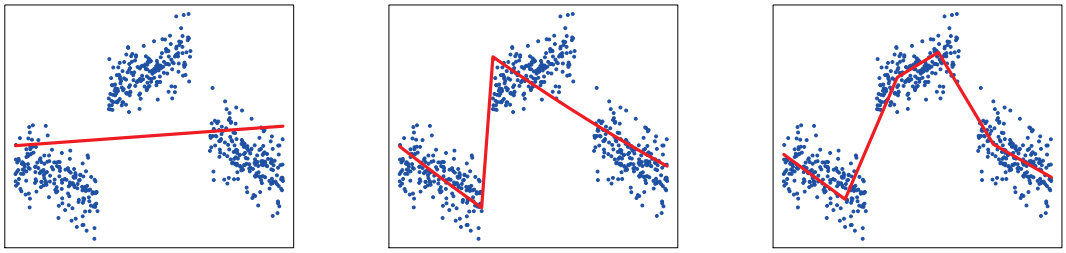
FIGURE 7. Results for synthetic data given in (i)-b with noise and outliers.

A data set is divided into training (80%) and test (20%) sets for each fold of the cross-validation. The results of algorithms are averages of the values obtained for 5 folds. We highlight the best results using the boldface font.

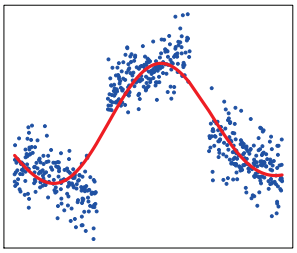
According to all five criteria, the A-PWLSVR algorithm is the best or the second best in six data sets: Airfoil self-noise, Red wine quality, BlogFeedback, SGEMM GPU kernel performance, Residential building, and Boston housing. Furthermore, the A-PWLSVR algorithm is the best concerning all but the MAE criterion in the Concrete compressive strength data set.



(A) A-PWLSVR (B) PWLREG (C) RFR



(D) MLR (E) NNR (F) MARS

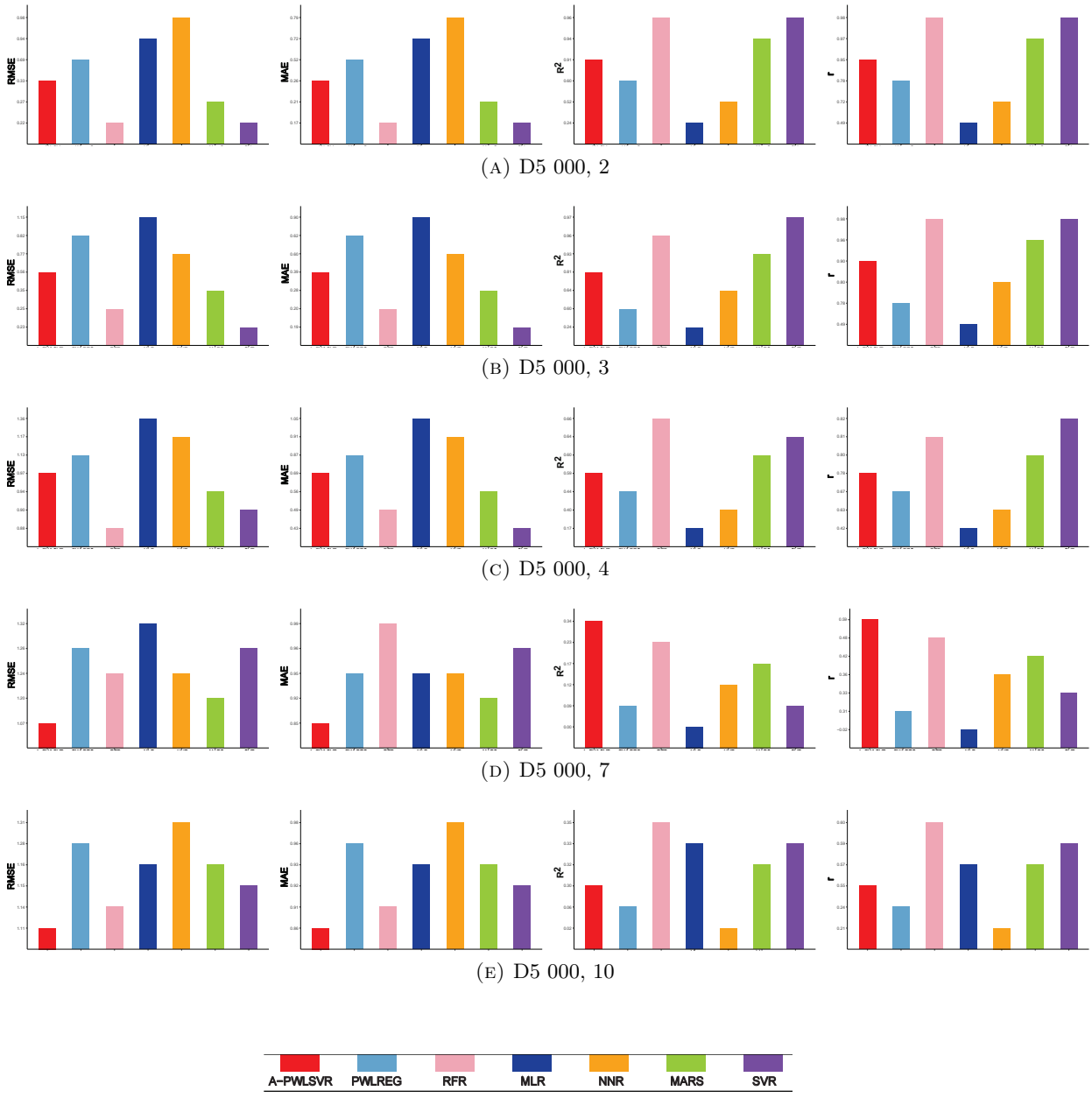


(G) SVR

FIGURE 8. Results for synthetic data given in (i)-a with clusterwise structure.

Overall, we can see the satisfactory performance of the A-PWLSVR algorithm in all data sets except for the Online news popularity data set. In this data set, the R^2-values by all algorithms, except MARS, are negative.

To further analysis of the results presented in Table 5, we draw a box plot for each performance measure. Recall that the smaller (larger) value of the median in a box plot indicates a better performance of the algorithm considering the criteria RMSE, MAE, and MAPE (R^2 and r). Since the values of the performance measures vary significantly for different data sets, we map the values obtained for the RMSE and MAE into the interval $[0, 1]$ and the negative values obtained



(A) D5 000, 2

(B) D5 000, 3

(C) D5 000, 4

(D) D5 000, 7

(E) D5 000, 10

FIGURE 9. Prediction performances of algorithms on synthetic data sets generated using regression function (ii).

for R^2 into the interval $[-1, -0.2]$ using the simple linear transformations. The mapping helps us with better visualization and presentation of the results.

It can be observed from Fig.10 that concerning the first two criteria (RMSE and MAE), A-PWLSVR reaches the lowest value of median and RFR has the second lowest. The A-PWLSVR algorithm has the smallest value of median in MAPE followed by SVR and RFR. The values of the median for both A-PWLSVR and RFR are very similar to R^2 and r. This figure shows that, in general, A-PWLSVR and RFR perform similarly and their performance is better than that of other algorithms.

Results of numerical experiments, presented in this section, demonstrate that the A-PWLSVR algorithm is especially accurate both as the approximation and prediction tool in data sets with

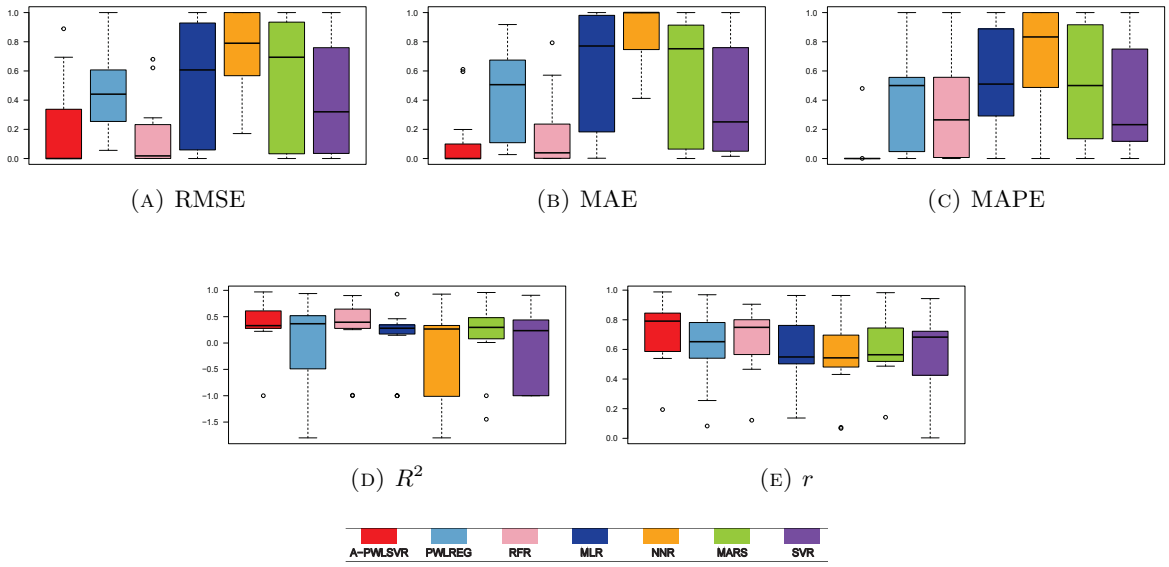
TABLE 4. Statistical significance tests (paired t-test) of performance difference between algorithms and baseline (MLR) on synthetic data generated using regression function (ii).

Measures	p-values between the algorithms and the baseline MLR					
	A-PWLSVR	PWLREG	RFR	NNR	MARS	SVR
RMSE	**0.023**	0.136	0.064	0.373	0.059	0.072
MAE	**0.030**	0.104	0.079	0.409	0.065	0.073
R^2	**0.031**	0.244	0.034	0.304	0.048	0.059
r	0.025	0.254	**0.013**	0.308	0.017	0.016

TABLE 5. Results obtained by different algorithms on real-world data sets

Algorithm	Residential	Boston	Concrete	Airfoil	Redwine	Whitewine	Combined	Online	Protein	BlogFeedback	SGEMM
						RMSE					
	$\times 10^4$	$\times 10^0$	$\times 10^1$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^5$	$\times 10^0$	$\times 10^2$	$\times 10^2$
A-PWLSVR	**0.015**	**5.422**	**0.927**	**3.830**	**0.656**	0.755	4.165	**0.113**	5.005	**0.218**	**1.939**
PWLREG	0.435	5.573	1.014	4.316	0.684	0.745	4.226	2.827	4.871	1.366	2.273
RFR	0.037	5.662	1.013	4.650	0.692	**0.720**	**3.349**	0.113	**3.561**	0.264	2.214
MLR	0.062	6.719	1.113	5.035	0.661	0.763	4.560	0.111	5.185	0.302	3.107
NNR	5.475	6.805	1.169	5.035	0.670	0.752	4.560	0.576	5.185	2.823	2.858
MARS	0.018	8.117	0.942	4.997	0.658	0.755	4.258	0.111	5.024	0.299	3.416
SVR	0.104	6.285	1.234	4.006	0.714	0.770	3.977	0.111	4.296	0.341	1.973
						MAE					
	$\times 10^4$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^3$	$\times 10^0$	$\times 10^2$	$\times 10^2$
A-PWLSVR	**0.009**	**4.070**	7.250	**2.915**	**0.503**	**0.569**	3.143	**2.530**	3.634	**0.041**	**1.119**
PWLREG	0.126	4.169	7.623	3.456	0.532	0.585	3.309	6.666	3.931	0.139	1.338
RFR	0.024	4.534	8.278	3.763	0.505	0.569	**2.430**	3.466	**2.521**	0.067	1.289
MLR	0.019	5.172	8.925	3.985	0.511	0.594	3.628	3.466	4.343	0.099	2.163
NNR	4.302	5.209	9.465	3.985	0.524	0.591	3.628	7.035	4.341	1.847	1.960
MARS	0.010	5.830	**6.698**	3.938	0.508	0.590	3.331	3.031	4.110	0.096	2.473
SVR	0.077	4.988	9.458	3.046	0.554	0.595	2.923	2.798	2.978	0.067	1.174
						MAPE					
	$\times 10^1$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^0$	$\times 10^6$	$\times 10^0$
A-PWLSVR	0.010	**0.230**	**0.260**	**0.023**	**0.092**	**0.099**	**0.006**	**0.838**	0.826	**0.000**	**0.720**
PWLREG	**0.005**	0.238	0.294	0.028	0.097	0.103	0.007	2.817	1.352	**0.000**	0.823
RFR	0.019	0.270	0.325	0.030	0.093	**0.099**	0.007	2.284	**0.341**	5.425	0.749
MLR	0.023	0.307	0.341	0.032	0.094	0.105	0.008	2.242	0.762	**0.000**	2.604
NNR	6.247	0.310	0.362	0.032	0.096	0.104	0.008	4.728	0.770	**0.000**	2.389
MARS	0.008	0.381	0.277	0.032	0.093	0.104	0.007	1.800	1.064	**0.000**	3.148
SVR	0.093	0.265	0.366	0.025	0.101	0.105	0.007	1.483	0.598	**0.000**	0.891
						R^2					
A-PWLSVR	**0.973**	0.490	**0.562**	**0.660**	**0.303**	0.263	**0.940**	-0.031	0.330	**0.294**	**0.222**
PWLREG	0.827	0.470	0.501	0.536	0.243	0.282	0.939	-8.7×10^3	0.366	-3.7×10	-0.373
RFR	0.849	**0.626**	0.395	0.511	0.301	**0.328**	0.901	-0.039	**0.661**	0.254	-0.394
MLR	0.194	0.287	0.461	0.398	0.290	0.245	0.928	-0.001	0.281	0.147	-9.610
NNR	-1.4×10^4	0.254	0.385	0.398	0.274	0.266	0.928	-2.6×10^2	0.281	-1.3×10^2	-7.241
MARS	0.959	-0.794	0.558	0.405	0.297	0.260	0.938	**0.010**	0.325	0.150	-1.8×10
SVR	-0.185	0.369	0.298	0.623	0.176	0.235	0.906	-0.003	0.507	-0.031	-0.379
						r					
A-PWLSVR	**0.988**	0.840	**0.838**	**0.850**	**0.574**	0.539	**0.969**	**0.194**	0.598	**0.640**	**0.791**
PWLREG	0.940	0.796	0.759	0.767	0.525	0.556	0.966	0.083	0.605	0.255	0.652
RFR	0.947	**0.866**	0.801	0.791	0.508	**0.591**	0.901	0.122	**0.818**	0.624	0.663
MLR	0.853	0.783	0.741	0.669	0.549	0.520	0.964	0.137	0.531	0.485	0.458
NNR	0.704	0.768	0.689	0.669	0.541	0.543	0.964	0.074	0.531	0.067	0.431
MARS	0.983	0.564	0.808	0.681	0.558	0.545	0.968	0.142	0.571	0.493	0.487
SVR	0.395	0.683	0.720	0.810	0.456	0.499	0.943	0.038	0.720	0.003	0.725

scattered data points (for example, Residential building and BlogFeedback data sets), in those with approximately piecewise linear structures, and also in data sets with clusterwise structures. In such data sets, this algorithm, in general, outperforms other regression algorithms. This observation justifies the development and application of the A-PWLSVR algorithm.



(A) RMSE (B) MAE (C) MAPE

(D) R^2 (E) r

FIGURE 10. Visualisation of results given in Table 5.

6. CONCLUSION

In this paper, a model for the piecewise linear regression problem and a novel algorithm for solving this problem are introduced. In the model, the regression function is estimated over the set of continuous piecewise linear functions represented as a maximum of the minima of linear functions. The regression estimate is defined using the L_1-risk and the piecewise linear regression problem is modeled by applying the support vector linear regression approach. This leads to the development of a model for piecewise linear regression: The piecewise linear support vector regression model. We consider two different versions of this model. The first version contains the flatness term and the second version is without the flatness term. In the first case, the objective function is a nonconvex piecewise quadratic function and in the second case, it is nonconvex piecewise linear.

The objective function is represented as a difference of two convex functions and the double bundle method for nonsmooth DC optimization is applied to minimize it. The new algorithm – called the adaptive piecewise linear support vector regression algorithm – is designed to solve the piecewise linear support vector regression problem. This algorithm incrementally constructs the piecewise linear estimator of the regression function starting from the linear function and stops adding pieces when the model is good enough. We present the proposed algorithm using both its flowchart and the step-by-step form.

The developed algorithm is evaluated as both the approximation and the prediction tool using synthetic and real-world data sets. In addition, it is compared with several mainstream machine learning regression algorithms using numerical results, various performance measures, the t-test, and box plots. Using the synthetic data sets, we demonstrate that the computational effort of the new algorithm increases close to quadratic depending on the number of input variables and roughly linearly depending on the number of data points. In some real-world data sets used in our experiments, the proposed algorithm is more time-consuming than other algorithms as it involves the solving of several DC optimization problems. However, the new algorithm is very robust to outliers and the obtained results demonstrate that it outperforms other regression algorithms in most data sets used in this paper. The new algorithm is especially accurate in scattered data sets and in data sets with piecewise linear and clusterwise structures.

7. Acknowledgments

The research was supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580) and by the Research Council of Finland (Project No. 289500, 319274, 345804, and 345805).

References

[1] Aggarwal, C. *Neural Networks and Deep Learning*, Springer, 2018, 520 p.
[2] An, L.T.H., Tao, P.D. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems, *Ann. Oper. Res.*, V.133, N.1, 2005, pp.23-46.
[3] Bagirov, A.M., Taheri, S., Karmitsa, N., Sultanova, N., Asadi, A. Robust piecewise linear L1-regression via nonsmooth DC optimization, *Optim. Methods Softw.*, V.37, N.4, 2022, pp.1289-1309.
[4] Bagirov, A.M., Taheri, S., Asadi, A. A difference of convex optimization algorithm for piecewise linear regression, *J. Ind. Manag. Optim.*, V.15, N.2, 2019, pp.909-932.
[5] Bagirov, A.M., Karmitsa, N., Mäkelä, M.M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*, Springer, Cham, 2014, 698 p.
[6] Bagirov, A.M., Clausen, C., Kohler, M. Estimation of a regression function by maxima of minima of linear functions, *IEEE Transactions on Information Theory*, V.55, N.2, 2009, pp.833-845.
[7] Bagirov, A.M., Clausen, C., Kohler, M. An L_2-boosting algorithm for estimation of a regression function, *IEEE Transactions on Information Theory*, V.56, N.3, 2010, pp.1417-1429.
[8] Bagirov, A.M., Clausen, C., Kohler, M. Kohler, An algorithm for the estimation of a regression function by continuous piecewise linear functions, *Comput. Optim. Appl.*, V.45, N.1, 2010, pp.159-179.
[9] Barati, M., Sharifian, S. A hybrid heuristic-based tuned support vector regression model for cloud load prediction, *J Supercomput. Journal*, V.71, N.11, 2015, pp.4235-4259.
[10] Breiman, L. Random forests, *Mach. Learn.*, V.45, N.1, 2001, pp.5-32.
[11] Chen, W., Hasanipanah, M., Nikafshan Rad, H., Danial Jahed, A., Tahir, M.M. A new design of evolutionary hybrid optimization of SVR model in predicting the blast-induced ground vibration, *Eng. Comput.,* V.37, N.2, 2021, pp.1455-1471.
[12] Cherkassky, V., Ma, Y. Practical selection of SVM parameters and noise estimation for SVM regression, *Neural Netw.*, V.17, N.1, 2004, pp.113-126.
[13] Clarke, F.H. *Optimization and Nonsmooth Analysis*, John Wiley & Sons, NY, USA, 1983, 321 p.
[14] Danandeh Mehr, A., Nourani, V., Karimi Khosrowshahi, V., Ghorbani, M.A. A hybrid support vector regression firefly model for monthly rainfall forecasting, *Int. J. Environ. Sci. Technol.*, V.16, N.1, 2019, pp.335-346.
[15] De Oliveira, W. The ABC of DC programming, *Set-Valued Var. Anal.*, V.28, N.4, 2020, pp.679-706.
[16] Dua, D. Graff, C. UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, <URL: `http://archive.ics.uci.edu/ml`> (12 December 2020), 2019.
[17] Friedman, J.H., Silverman, B.W. Flexible parsimonious smoothing and additive modeling, *Technometrics*, V.31, N.1, 1989, pp.3-21.
[18] Friedman, J.H. Multivariate adaptive regression splines (with discussion), *Ann. Stat.*, V.19, N.1, 1991, pp.79-141.
[19] Friedman, J.H., Hastie, T., Tibshirani, R. *The Elements of Statistical Learning*, Springer, Berlin, 2001, 745 p.
[20] Gorokhovik, V.V., Zorko, O.I. Birkhoff, G. Piecewise affine functions and polyhedral sets, *Optim*, V.31, N.3, 1994, pp.209-221.
[21] Horst, R. Thoai, N.V. DC programming: overview, *J. Optimiz. Theory App.*, V.103, N.1, 1999, pp.319-350.
[22] Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S. Double bundle method for finding Clarke stationary points in nonsmooth DC programming, *SIAM J. Optimiz. Journal*, V.28, N.2, 2018, pp.1892-1919.
[23] Kiwiel, K.C. Proximity control in bundle methods for convex nondifferentiable minimization, *Math. Program.*, V.46, N.1, 1990, pp.105-122.
[24] Kwok, J.T., Tsang, I.W. Linear dependency between /spl epsi/ and the input noise in /spl epsi/-support vector regression, *IEEE T Neural Networ.*, V.14, N.3, 2003, pp.544-553.
[25] Le, V.L., Lauer, F., Bako, L., Bloch, G. Learning nonlinear hybrid systems: from sparse optimization to support vector regression, *16th International Conference on Hybrid systems: computation and control, HSCC '13*, Philadelphia, United States, 2013, pp.33-42.
[26] Li, Y. Research on multi-core learning SVM algorithm and identification of pulmonary nodules, *College of Communication Engineering, Jilin University, Jilin, China*, 2014.

[27] Liaw, A., Wiener, M. Breiman and Cutler's random forests for classification and regression, Package "random forest", <URL: `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`> (26 November 2021), 2018, 29 p.
[28] Liua, S., Tai, H., Ding, Q., Li, D., Xu, L., Wei, Y. A hybrid approach of support vector regression with genetic algorithm optimization for aquaculture water quality prediction, *Math. Comput. Model.* , V.58, N.3-4, 2013, pp.458-465.
[29] Luo, X., Yuan, X., Zhu, S., Xu, Z., Meng, L., Peng, J. A hybrid support vector regression framework for streamflow forecast, *J. Hydrol.*, V.568, 2019, pp.184-193.
[30] Melzer, D. On the expressibility of piecewise-linear continuous functions as the difference of two piecewise-linear convex functions, *Quasidifferential Calculus*, Springer, Berlin, Heidelberg, 1986, pp.118-134.
[31] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.C., Lin, C.C. Misc Functions of the Department of Statistics, Probability Theory Group, TU Wien, Package "e1071", <URL: `http://cran.r-project.org/web/packages/e1071/e1071.pdf`> (12 December 2020), 2017, 67 p.
[32] Milborrow, S. Multivariate Adaptive Regression Splines, Package "earth", <URL: `http://cran.r-project.org/web/packages/earth/earth.pdf`> (12 December 2020), 2017, 53 p.
[33] Mu, T., Nandi, A.K. Automatic tuning of L2-SVM parameters employing the extended Kalman filter, *Expert. Syst. Appl.*, V.26, N.2, 2009, pp.160-175.
[34] Nasir, M., Mysorewala, M., Cheded, L., Siddiqui, B., Sabih, M. Measurement error sensitivity analysis for detecting and locating leak in pipeline using ANN and SVM, *IEEE 11th International Multi-Conference on Systems, Signals and Devices, SSD,* 2014.
[35] Ripley, B., Venables, W. Feed-Forward Neural Networks and Multinomial Log-Linear Models, Package "nnet", <URL: `http://cran.r-project.org/web/packages/nnet/nnet.pdf`> (12 December 2020), 2016, 11 p.
[36] Shao, Z., Er, M.J., Wang, N. An effective semi-cross-validation model selection method for extreme learning machine with ridge regression, *Neurocomputing*, V.151, N.2, 2015, pp.933-942.
[37] Smola, A.J., Schölkopf, B. A tutorial on support vector regression, *Stat. Comput.*, V.14, N.3, 2004, pp.199-222.
[38] Sopyla, K., Drozda, P. Stochastic gradient descent with Barzilai-Borwein update step for SVM, *Inf. Sci.*, V.316, 2015, pp.218-233.
[39] Steinwart, I., Christmann, A. *Support Vector Machines*, Springer Science & Business Media, 2008, 603 p.
[40] Tao, P. D., An, L.T.H. Convex analysis approach to DC programming: theory, algorithms and applications, *Acta Math. Vietnam.*, V.22, N.1, 1997, pp.289-355.
[41] Tuy, H. *Convex Analysis and Global Optimization, Nonconvex Optimization and Its Applications*, Kluwer, Dordrecht, 1998, 22 p.
[42] Van Ackooij, W., de Oliveira, W. Nonsmooth and nonconvex optimization via approximate difference-of-convex decompositions, *J. Optimiz. Theory App.*, V.182, N.1, 2019, pp.49-80.
[43] Wang, H., Xu, D. Parameter selection method for support vector regression based on adaptive fusion of the mixed kernel function, *J. Control Sci. Eng*, V.2017, 2017, Article No.3614790, 12 p.
[44] Wilkinson, G.N., Rogers, C.E. Symbolic description of factorial models for analysis of variance, *J. R. Stat. Soc.,* V.22, N.3, 1973, pp.392-399.
[45] Zhu, W., Song, Y., Xiao, Y. A new support vector machine plus with pinball loss, *J. Classif.,* V.35, N.1, 2018, pp.52-70.

Adil M. Bagirov - graduated from Faculty of Applied Mathematics, Baku State University in 1983. He received his Ph.D. degree in Mathematical Cybernetics from the Institute of Cybernetics of Azerbaijan National Academy of Sciences in 1983. Presently, he is a Professor at Federation University Australia, Ballarat, Victoria, Australia. His research interests are on nonsmooth and global optimization and their applications.

Sona Taheri - graduated from Tabriz University in 2003. She received her Ph.D. degree in Information Technology from Federation University Australia in 2012. Presently, she is a Lecturer at RMIT University Australia, Melbourne, Victoria, Australia. Her research interests are on nonsmooth optimization, machine learning and their applications.

Napsu Karmitsa - graduated from University of Jyväskylä in 1998. She received her Ph.D. in Scientific Computing from University of Jyväskylä in 2004. Presently, she is a Senior Research Fellow at University of Turku, Finland. Her research interests are on nonsmooth optimization and data sciences.

Kaisa Joki - graduated from University of Turku in 2013. She received her Ph.D. in Mathematical Modelling from University of Turku in 2018. Presently, she is a Lecturer at University of Turku, Finland. Her research interests are on nonsmooth and multiobjective optimization and their applications.

Marko M. Mäkelä - graduated from University of Jyväskylä in 1986. He received his Ph.D. in Computer Science from from University of Jyväskylä in 1990. Presently, he is a Professor at University of Turku, Finland. His research interests are on nonsmooth analysis, nonsmooth and multiobjective optimization.