Joki, K., Bagirov, A. M., Karmitsa, N., Mäkelä, M. M., & Taheri, S. (2020). Clusterwise support vector linear regression. *European Journal of Operational Research, 287*(1), 19–35.

Which has been published in final form at:
https://doi.org/10.1016/j.ejor.2020.04.032

Clusterwise support vector linear regression

Kaisa Joki[a,*], Adil M. Bagirov[b], Napsu Karmitsa[a], Marko M. Mäkelä[a], Sona Taheri[b]

[a] *Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland*
[b] *School of Science, Engineering and Information Technology, Federation University Australia, University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia*

Abstract

In clusterwise linear regression (CLR), the aim is to simultaneously partition data into a given number of clusters and to find regression coefficients for each cluster. In this paper, we propose a novel approach to model and solve the CLR problem. The main idea is to utilize the support vector machine (SVM) approach to model the CLR problem by using the SVM for regression to approximate each cluster. This new formulation of the CLR problem is represented as an unconstrained nonsmooth optimization problem, where we minimize a difference of two convex (DC) functions. To solve this problem, a method based on the combination of the incremental algorithm and the double bundle method for DC optimization is designed. Numerical experiments are performed to validate the reliability of the new formulation for CLR and the efficiency of the proposed method. The results show that the SVM approach is suitable for solving CLR problems, especially, when there are outliers in data.

Keywords: Data mining, Nonsmooth optimization, Clusterwise linear regression, DC optimization, Bundle methods

1. Introduction

Clusterwise linear regression (CLR) is a technique for fitting multiple hyperplanes to mutually exclusive subsets of observations of a data set (Späth, 1979). It is a combination of two techniques: clustering and regression. Applications

*Corresponding author
Email addresses: kaisa.joki@utu.fi (Kaisa Joki), a.bagirov@federation.edu.au (Adil M. Bagirov), napsu@karmitsa.fi (Napsu Karmitsa), makela@utu.fi (Marko M. Mäkelä), s.taheri@federation.edu.au (Sona Taheri)

of CLR include, for example, the consumer benefit segmentation (Wedel and Kistemaker, 1989), market segmentation (Preda and Saporta, 2005), modeling of the metal inert gas welding process (Ganjigatti et al., 2007), rainfall prediction (Bagirov et al., 2017) and PM10 prediction (Poggi and Portier, 2005).

To date, various models of the CLR problem have been proposed and different algorithms have been developed based on them. A nonlinear programming formulation of the CLR problem is proposed in Lau et al. (1999). In this model the objective function is nonconvex quadratic and all variables are continuous. The number of variables depends on the number of linear functions, input variables and data points in a data set and, thus, becomes prohibitively large in large data sets.

A mixed-integer linear programming formulation for the CLR problem with the sum of the absolute error as the objective is introduced in Bertsimas and Shioda (2007). In this approach, the data set is first divided into a small number of clusters, and a mixed-integer programming algorithm is applied to approximate them. In this model the number of variables depends on the number of linear functions, input variables and data points in a data set and, therefore, it contains a large number of binary variables in large data sets.

A mixed logical-quadratic programming formulation of CLR is presented in Carbonneau et al. (2011). An approach based on a quadratic mixed-integer program and a set partition formulation is proposed to model CLR by Park et al. (2017). A model based on the combination of fuzzy clustering and fuzzy regression is considered in D'Urso et al. (2010). In addition, several mixture models have been developed for CLR problems in DeSarbo and Cron (1988); García-Escudero et al. (2010).

A nonsmooth optimization model is developed in Bagirov et al. (2013) and a nonsmooth difference of convex (DC) optimization model is introduced in Bagirov and Ugon (2018). These models contain only continuous variables. In addition, the number of variables depends only on the number of linear functions and the number of input variables. Therefore, the number of variables is significantly less than that in models based on nonlinear programming, mixed-integer linear and quadratic mixed-integer programming techniques.

Algorithms for solving the CLR problem include also those which are ex-

tensions of clustering algorithms such as the k-means (Späth, 1979) and the expectation-maximization algorithms (EM) (Gaffney and Smyth, 1999) and those based on the nonlinear programming (Lau et al., 1999), the mixed integer linear programming (Bertsimas and Shioda, 2007), the mixed integer nonlinear programming (Carbonneau et al., 2012; DeSarbo et al., 1989), nonsmooth optimization (Bagirov et al., 2013, 2015a,b; Bagirov and Ugon, 2018) and mixture models (DeSarbo and Cron, 1988; García-Escudero et al., 2010).

In this paper, a new approach for modelling and solving CLR problems is proposed using support vector machines (SVM) for regression (Collobert and Bengio, 2001; Smola and Schölkopf, 2004). By applying the SVM formulation for regression, the CLR problem is modelled as a constrained nonsmooth optimization problem. Then using the penalty function this problem is replaced by an unconstrained nonsmooth optimization problem, where the regression errors are defined using the L_1-risk and small perturbations from hyperplanes are tolerated without penalty. This model differs from the typical nonsmooth nonconvex formulation of CLR, where regression errors are defined using the L_2-risk and all deviations are penalized. In particular, the model and the solution approach are different from those given in Bagirov and Ugon (2018). First, in this paper we use L_1-risk whereas in Bagirov and Ugon (2018) the L_2-risk is applied. Second, the objective function in the new model is piecewise linear while it is piecewise quadratic in the model proposed in (Bagirov and Ugon, 2018).

To solve the CLR problem using its new formulation, we design an algorithm based on the combination of the incremental algorithm and the bundle-type method. Since the objective function in this formulation is represented as a difference of two convex (DC) functions, the proposed algorithm uses the double bundle method (DBDC) (Joki et al., 2018) developed for nonsmooth DC optimization. This enables us to utilize the DC structure. However, the DBDC is a local method. Therefore, to improve the quality of the obtained solution, the DBDC is combined with an incremental approach introduced in Bagirov et al. (2013). The incremental approach allows us to generate starting points which are rough estimates of the solution of the CLR problem. This way we are able to design a more accurate algorithm for solving the nonconvex CLR problem.

In addition, solutions to the intermediate CLR problems with a smaller number of hyperplanes are obtained as the by-products from the incremental approach. The proposed algorithm is tested using some synthetic and real-world data sets for regression to validate the adequacy of the new SVM based formulation for the CLR problem.

The rest of the paper is organized as follows. Section 2 provides some preliminaries. The SVM reformulations of the CLR and auxiliary CLR problems are given in Section 3. Section 4 presents the new method DBDC-CSVLR. Numerical results are reported in Section 5 and Section 6 contains some concluding remarks.

2. Preliminaries

We start with some definitions and results from nonsmooth analysis and DC optimization. For more details we refer to Bagirov et al. (2014); Clarke (1983); Horst and Thoai (1999); Le Thi and Pham Dinh (2005); Pham Dinh and Le Thi (1997); Strekalovsky (2015); Tuy (1998).

We denote by $\mathbb{R}^n$ the n-dimensional Euclidean space. The inner product is denoted by $\boldsymbol{u}^T\boldsymbol{v} = \sum_{i=1}^n u_i v_i$, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ and the associated norm by $\|\boldsymbol{u}\| = (\boldsymbol{u}^T\boldsymbol{u})^{1/2}$. The set $B(\boldsymbol{x};\varepsilon) = \{\boldsymbol{y} \in \mathbb{R}^n \,|\, \|\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon\}$ is the open ball centered at $\boldsymbol{x}$ with the radius $\varepsilon > 0$. The notation "conv" is used for a convex hull of a set and "cl" for a closure of a set.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Its *subdifferential* at $\boldsymbol{x} \in \mathbb{R}^n$ is given by (Rockafellar, 1970)

$$\partial_c f(\boldsymbol{x}) = \left\{\boldsymbol{\xi} \in \mathbb{R}^n | \, f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) \;\text{ for all } \boldsymbol{y} \in \mathbb{R}^n\right\}$$

being a nonempty, convex and compact set. For convex functions, we have some useful subdifferential calculus rules. The following lemma presents two of them (for proofs see, e.g., Bagirov et al. (2014)).

Lemma 2.1. *Let functions $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, k$ be convex. Then*

(i) the function $g(\boldsymbol{x}) = \sum_{i=1}^{k} f_i(\boldsymbol{x})$ *is convex and its subdifferential is*

$$\partial_c g(\boldsymbol{x}) = \sum_{i=1}^{k} \partial_c f_i(\boldsymbol{x});$$

(ii) the function $h(\boldsymbol{x}) = \max\{\, f_i(\boldsymbol{x}) \mid i = 1, \ldots, k\}$ *is convex and its subdifferential is*

$$\partial_c h(\boldsymbol{x}) = \operatorname{conv}\{\, \partial_c f_i(\boldsymbol{x}) \mid i \in I(\boldsymbol{x})\},$$

where $I(\boldsymbol{x}) = \{\, i \in \{1, \ldots, k\} \mid f_i(\boldsymbol{x}) = h(\boldsymbol{x})\}$.

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

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

For a locally Lipschitz function f, the *generalized directional derivative* at a point $\boldsymbol{x} \in \mathbb{R}^n$ with respect to a direction $\boldsymbol{d} \in \mathbb{R}^n$ is (Clarke, 1983)

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

and the *generalized subdifferential* $\partial f(\boldsymbol{x})$ at $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \; f^\circ(\boldsymbol{x}; \boldsymbol{d}) \geq \boldsymbol{\xi}^T \boldsymbol{d} \quad \text{for all } \boldsymbol{d} \in \mathbb{R}^n \right\}.$$

Each vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a *subgradient*. Since $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$ holds for convex functions (Clarke, 1983) we will use the notation ∂f also for subdifferentials of convex functions.

The *Goldstein ε-subdifferential* of a locally Lipschitz function f with $\varepsilon \geq 0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is (Mäkelä and Neittaanmäki, 1992)

$$\partial_\varepsilon^G f(\boldsymbol{x}) = \operatorname{cl} \operatorname{conv}\{\, \partial f(\boldsymbol{y}) \,|\, \boldsymbol{y} \in B(\boldsymbol{x}; \varepsilon)\}.$$

This subdifferential is an extension of $\partial f(\boldsymbol{x})$ since $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for all $\varepsilon \geq 0$ and $\partial_0^G f(\boldsymbol{x}) = \partial f(\boldsymbol{x})$.

Definition 2.2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *DC* if it can be represented as a difference of two convex functions $f^1, f^2 : \mathbb{R}^n \to \mathbb{R}$ in the form

$$f(\boldsymbol{x}) = f^1(\boldsymbol{x}) - f^2(\boldsymbol{x}).$$

Here, $f^1 - f^2$ is a *DC decomposition* of f and convex functions f^1 and f^2 are called *DC components*. DC functions are locally Lipschitz and typically nonconvex. If f is nonsmooth, then at least one of the DC components is nonsmooth. In addition, DC functions preserve the DC structure under some simple operations frequently used in optimization as the following lemma demonstrates.

Lemma 2.3. *(Tuy, 1998) Let $f_i = f_i^1 - f_i^2$ for $i = 1, \ldots, k$ be DC functions. Then*

(i) $g(\boldsymbol{x}) = \min\{ f_i(\boldsymbol{x}) \mid i = 1, \ldots, k\}$ is a DC function and its DC decomposition $g = g^1 - g^2$ can be written with the DC components

$$g^1(\boldsymbol{x}) = \sum_{i=1}^{k} f_i^1(\boldsymbol{x}) \quad \textit{and}$$
$$g^2(\boldsymbol{x}) = \max_{i=1,\ldots,k} \left\{ f_i^2(\boldsymbol{x}) + \sum_{j=1, j\neq i}^{k} f_j^1(\boldsymbol{x}) \right\};$$

(ii) $h(\boldsymbol{x}) = \max\{ f_i(\boldsymbol{x}) \mid i = 1, \ldots, k\}$ is a DC function and its DC decomposition $h = h^1 - h^2$ can be written with the DC components

$$h^1(\boldsymbol{x}) = \max_{i=1,\ldots,k} \left\{ f_i^1(\boldsymbol{x}) + \sum_{j=1, j\neq i}^{k} f_j^2(\boldsymbol{x}) \right\} \quad \textit{and}$$
$$h^2(\boldsymbol{x}) = \sum_{i=1}^{k} f_i^2(\boldsymbol{x}).$$

An unconstrained DC programming problem is formulated as

$$\begin{cases} \min & f(\boldsymbol{x}) = f^1(\boldsymbol{x}) - f^2(\boldsymbol{x}) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{1}$$

For a point $\boldsymbol{x}^* \in \mathbb{R}^n$ to be a local minimizer of the problem (1), it is necessary that $\partial f^2(\boldsymbol{x}^*) \subseteq \partial f^1(\boldsymbol{x}^*)$. 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. Therefore, in most algorithms the following weaker necessary conditions are used:

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) \quad \text{(Clarke stationarity)}$$

and

$$\partial f^1(\boldsymbol{x}^*) \cap \partial f^2(\boldsymbol{x}^*) \neq \emptyset \quad \text{(criticality)}.$$

It is known that any Clarke stationary point is also critical. However, the opposite claim is not always true (Joki et al., 2018).

3. SVM based clusterwise linear regression

In this section, we introduce a new model of the clusterwise linear regression (CLR) problem by applying the support vector machine (SVM) approach for regression. This differs from the SVM for general regression problems since we consider the estimation of several regression functions instead of only one. The new SVM based formulation of CLR enables us to ignore small perturbations and this makes it different from the typical nonsmooth nonconvex CLR formulation. In addition, we use the L_1-risk to compute regression errors whereas the existing CLR formulations mostly apply the L_2-risk. With this selection our model is less sensitive to outliers.

The new formulation of the CLR problem is called CSVLR (Clusterwise Support Vector Linear Regression). Furthermore, we introduce the auxiliary CSVLR problem used to find starting points which are rough estimates of the solution of the original CSVLR problem. The SVM for general regression problems with one linear function is discussed in Collobert and Bengio (2001); Smola and Schölkopf (2004).

Suppose that we are given a finite data set

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

where $\boldsymbol{a}^i$ is an input and b_i is its output. The aim of the CLR is twofold: the

data set A is partitioned into k clusters and at the same time each cluster is approximated by one linear function. To achieve this goal, we need to optimize the overall fit. It is worth noting that the number of clusters k needs to be defined by the user before solving the CLR problem.

In what follows, let A^j for $j = 1, \ldots, k$ be nonempty *clusters* such that

$$A^j \bigcap A^l = \emptyset,\ j, l = 1, \ldots, k,\ l \neq j \ \text{ and } \ A = \bigcup_{j=1}^{k} A^j,$$

and $\{\boldsymbol{x}^j, y_j\}$ be linear *regression coefficients* computed using solely data points from the cluster A^j, $j = 1, \ldots, k$. For a given data point $(\boldsymbol{a}^i, b_i) \in A$ and a coefficient $\{\boldsymbol{x}^j, y_j\}$ the general form of the regression error can be defined as

$$\left((\boldsymbol{x}^j)^T \boldsymbol{a}^i + y_j - b_i\right)^p \quad \text{for } p \geq 1.$$

By selecting $p = 1$ we get the L_1*-risk*. In the case of $p = 2$, the regression error is defined using the L_2*-risk*.

3.1. SVM approach to linear regression

We start with the brief description of the SVM approach for linear regression, where the aim is to approximate the given data set A using one hyperplane f (linear function) with a precision $\varepsilon > 0$. The parameter ε is fixed by the user and it typically depends on data. In ε-SVM linear regression we define the function f as follows (Collobert and Bengio, 2001; Smola and Schölkopf, 2004)

$$f(\boldsymbol{a}) = \boldsymbol{x}^T \boldsymbol{a} + y,$$

and try to determine the regression coefficients $\boldsymbol{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}$ in such a way that for each point $(\boldsymbol{a}^i, b_i) \in A$ the deviation between $f(\boldsymbol{a}^i)$ and the actually obtained target b_i is at most ε. Moreover, the function f is required to be as flat as possible meaning that the smaller the norm of $\boldsymbol{x}$ is the better. With this requirement we can reduce the complexity of the regression problem when there are large number of input variables. In addition, the flatness condition guarantees that the problem has a unique solution.

The SVM for regression can be formulated as the following nonsmooth con-

vex optimization problem

$$\begin{cases} \min & \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{s.t.} & |\boldsymbol{x}^T\boldsymbol{a}^i + y - b_i| \leq \varepsilon, \quad i = 1, \ldots, m \\ & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}. \end{cases} \tag{2}$$

The existence of a solution for the problem (2) requires the existence of the hyperplane f approximating all points $(\boldsymbol{a}^i, b_i) \in A$ with the precision ε. Since this requirement is not always possible to fulfill in practice it is often more convenient to relax the constraints to achieve feasibility. By introducing a regularization parameter $C > 0$ and applying the penalty function approach, the problem (2) can be reformulated as an unconstrained convex nonsmooth optimization problem

$$\begin{cases} \min & \frac{1}{2}\|\boldsymbol{x}\|^2 + C\sum\limits_{i=1}^{m} \max\left\{0, \left|\boldsymbol{x}^T\boldsymbol{a}^i + y - b_i\right| - \varepsilon\right\} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}. \end{cases} \tag{3}$$

This formulation gives us more "freedom" in the solution process since all points $\boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}$ are feasible. In practice, most data sets contain some noise and, due to this, it is worthwhile to allow small perturbations from the hyperplanes.

3.2. Nonsmooth formulation of CLR

In CLR, we want to approximate the data set A using k hyperplanes denoted by $\{\boldsymbol{x}^1, y_1\}, \ldots, \{\boldsymbol{x}^k, y_k\}$ where $\boldsymbol{x}^i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$. The nonsmooth formulation of CLR is typically modelled with the piecewise quadratic fit function (Bagirov et al. (2013, 2015a); Karmitsa et al. (2016)) and has the form

$$\begin{cases} \min & \hat{F}_k(\boldsymbol{x}, \boldsymbol{y}) = \sum\limits_{i=1}^{m} \min\limits_{j=1,\ldots,k} \left\{((\boldsymbol{x}^j)^T\boldsymbol{a}^i + y_j - b_i)^2\right\} \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{R}^{nk},\ \boldsymbol{y} \in \mathbb{R}^k, \end{cases} \tag{4}$$

which in the following is called PQ-CLR (Piecewise Quadratic CLR model). It is worth noting that the model is nonconvex when $k > 1$ and utilizes the L_2-risk to determine regression errors. It is also possible to replace the L_2-risk with the L_1-riskin (4). However, with the L_2-risk the regression error is smooth and

the objective function has better (sub)differentiability properties than with the L_1-risk. For these reasons, the PQ-CLR problem is considered to be easier and simpler to solve than its variation with the L_1-risk, although with the L_1-riskthe problem (4) is significantly less sensitive to outliers.

3.3. Formulation of CSVLR problem

In CSVLR, we are looking for k hyperplanes to approximate the data set A with a precision $\varepsilon > 0$. The regression coefficients of these hyperplanes are denoted by $\{\boldsymbol{x}^1, y_1\}, \ldots, \{\boldsymbol{x}^k, y_k\}$ where $\boldsymbol{x}^i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$ and, in what follows, the vectors $\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k)^T \in \mathbb{R}^{nk}$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_k)^T \in \mathbb{R}^k$ are the combined representation of these coefficients. Moreover, a point $(\boldsymbol{a}^i, b_i) \in A$ is associated with the hyperplane providing the smallest regression error. In the spirit of (2), the *CSVLR problem* can be formulated as follows

$$\begin{cases} \min & \frac{1}{2} \sum\limits_{j=1}^{k} \|\boldsymbol{x}^j\|^2 \\ \text{s. t.} & \min\limits_{j=1,\ldots,k} \left|(\boldsymbol{x}^j)^T \boldsymbol{a}^i + y_j - b_i\right| \leq \varepsilon, \ i = 1, \ldots, m \\ & \boldsymbol{x} \in \mathbb{R}^{nk}, \ \boldsymbol{y} \in \mathbb{R}^k. \end{cases}$$

By utilizing the same strategy as in Subsection 3.1, we are able to write the unconstrained nonsmooth version of the CSVLR problem as

$$\begin{cases} \min & F_k(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^{nk}, \ \boldsymbol{y} \in \mathbb{R}^k \end{cases} \tag{5}$$

with the objective function

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C \sum_{i=1}^{m} \max\left\{0, \psi_i(\boldsymbol{x}, \boldsymbol{y})\right\},$$

where

$$\psi_i(\boldsymbol{x}, \boldsymbol{y}) = \min_{j=1,\ldots,k} \left|(\boldsymbol{x}^j)^T \boldsymbol{a}^i + y_j - b_i\right| - \varepsilon, \ i = 1, \ldots, m. \tag{6}$$

It is worth noting that unlike (3) the problem (5) is nonconvex for $k > 1$ since functions ψ_i are nonconvex. In addition, the flatness of hyperplanes is expressed

as the sum of the squared norms of coefficients $\boldsymbol{x}^j$.

As mentioned in Subsection 3.1 flatness of a linear function is used to reduce the complexity of the problem when there are a large number of input variables. Usually, in SVM for regression the large number of variables appears when kernels are applied. Since we do not apply any kernels the number of variables is not large and therefore, flatness of coefficients $\boldsymbol{x}^j$ is not required. Furthermore, for the nonconvex problem (5) the flatness condition cannot guarantee the uniqueness of the solution. For these reasons, we remove the quadratic terms from the function F_k and rewrite the problem (5) as

$$\begin{cases} \min & F_k(\boldsymbol{x}, \boldsymbol{y}) = \sum\limits_{i=1}^{m} \max\{0, \psi_i(\boldsymbol{x}, \boldsymbol{y})\} \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^{nk},\ \boldsymbol{y} \in \mathbb{R}^k. \end{cases} \tag{7}$$

Note that the objective function in (7) is not subdifferentially regular and, therefore, the calculation of its subgradients is not always an easy task. However, the objective F_k is a DC function and subgradients of its DC components can be efficiently calculated.

Next, we give a DC representation for the function F_k. To simplify the notations, we denote by

$$e_i(\boldsymbol{x}^j, y_j) = \left|(\boldsymbol{x}^j)^T \boldsymbol{a}^i + y_j - b_i\right|$$

the error of the point $(\boldsymbol{a}^i, b_i) \in A$ from the hyperplane with the regression coefficients $\{\boldsymbol{x}^j, y_j\}$.

Proposition 3.1. *The function F_k defined in* (7) *is DC and its DC decomposition is*

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = F_k^1(\boldsymbol{x}, \boldsymbol{y}) - F_k^2(\boldsymbol{x}, \boldsymbol{y}),$$

where the DC components are

$$F_k^1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \max\left\{ \sum_{j=1}^{k} e_i(\boldsymbol{x}^j, y_j), \max_{j=1,\ldots,k} \sum_{t=1, t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon \right\}$$

and

$$F_k^2(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m}\left(\max_{j=1,\ldots,k}\sum_{t=1,t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon\right).$$

Proof. Consider the function $\psi_i(\boldsymbol{x},\boldsymbol{y})$ for $i = 1,\ldots,m$ defined in (6). The first term in this function is a minimum of convex piecewise linear functions and, thus, it is of the form presented in the case (i) of Lemma 2.3. Since the function e_i is convex we get

$$\psi_i(\boldsymbol{x},\boldsymbol{y}) = \psi_i^1(\boldsymbol{x},\boldsymbol{y}) - \psi_i^2(\boldsymbol{x},\boldsymbol{y})$$

with the DC components

$$\psi_i^1(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{k} e_i(\boldsymbol{x}^j, y_j) \quad \text{and}$$
$$\psi_i^2(\boldsymbol{x},\boldsymbol{y}) = \max_{j=1,\ldots,k}\sum_{t=1,t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon.$$

Both functions ψ_i^1 and ψ_i^2 are piecewise linear and convex since ψ_i^1 is a sum of convex piecewise linear functions and ψ_i^2 is a maximum of convex piecewise linear functions.

Next, consider the function

$$\varphi_i(\boldsymbol{x},\boldsymbol{y}) = \max\{0, \psi_i(\boldsymbol{x},\boldsymbol{y})\} \quad \text{for } i = 1,\ldots,m,$$

which can be represented as a difference of convex functions φ_i^1 and φ_i^2 by utilizing the case (ii) of Lemma 2.3:

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

where

$$\varphi_i^1(\boldsymbol{x},\boldsymbol{y}) = \max\left\{\psi_i^1(\boldsymbol{x},\boldsymbol{y}), \psi_i^2(\boldsymbol{x},\boldsymbol{y})\right\} \quad \text{and}$$
$$\varphi_i^2(\boldsymbol{x},\boldsymbol{y}) = \psi_i^2(\boldsymbol{x},\boldsymbol{y}).$$

Thus, the function F_k can be rewritten as

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

and its DC representation is

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = F_k^1(\boldsymbol{x}, \boldsymbol{y}) - F_k^2(\boldsymbol{x}, \boldsymbol{y})$$

with the DC components

$$F_k^1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \varphi_i^1(\boldsymbol{x}, \boldsymbol{y}) \quad \text{and}$$
$$F_k^2(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \varphi_i^2(\boldsymbol{x}, \boldsymbol{y}).$$

This completes the proof. □

Note that a DC function has an infinite number of different DC decompositions and the DC decomposition of the function F_k presented in Proposition 3.1 is only one option from the set of possible representations. For example, another DC decomposition is presented in Joki et al. (2017b), where the authors show using some preliminary testing that there is no real difference between the results obtained by that DC decomposition and the one given in Proposition 3.1.

3.4. Auxiliary CSVLR problem

The problem (7) is nonconvex and it may have many local solutions. The success of local search methods in finding high quality solutions to this problem strongly depends on the choice of starting points. Therefore, it is imperative to use a special procedure to generate such points.

In order to design such a procedure, we apply an incremental approach. This approach is similar to that introduced in Bagirov et al. (2013), but instead of CLR problems, it is designed for CSVLR problems. The basic idea in the incremental approach is that the solution for the CSVLR problem with $k-1$ hyperplanes can be used to derive good starting points for the CSVLR problem with k hyperplanes. The concept of the auxiliary problem has a central role in the incremental approach.

Let $(\boldsymbol{x}^1, y_1, \boldsymbol{x}^2, y_2, \ldots, \boldsymbol{x}^{k-1}, y_{k-1})$ be the (global) solution to the CSVLR problem (7) with $k-1$ hyperplanes. The *regression error* of the data point $(\boldsymbol{a}^i, b_i) \in A$ is denoted by

$$r_{k-1}^i = \max\Big\{0, \min_{j=1,\ldots,k-1} e_i(\boldsymbol{x}^j, y_j) - \varepsilon\Big\}.$$

The *kth auxiliary CSVLR problem* is

$$\begin{cases} \min & \bar{F}_k(\boldsymbol{u}, v) \\ \text{s.t.} & \boldsymbol{u} \in \mathbb{R}^n,\ v \in \mathbb{R} \end{cases} \tag{8}$$

with the objective function

$$\bar{F}_k(\boldsymbol{u}, v) = \sum_{i=1}^{m} \min\Big\{r_{k-1}^i,\ \phi_i(\boldsymbol{u}, v)\Big\}, \tag{9}$$

where

$$\phi_i(\boldsymbol{u}, v) = \max\big\{0,\ e_i(\boldsymbol{u}, v) - \varepsilon\big\}.$$

Notice that if $r_{k-1}^i = 0$ for $(\boldsymbol{a}^i, b_i) \in A$, then this point $(\boldsymbol{a}^i, b_i)$ can be omitted from the problem (8). Thus, the only interesting points in the auxiliary problem are those for which $r_{k-1}^i > 0$. In addition, the auxiliary CSVLR problem (8) is much easier and less time-consuming to solve than the original CSVLR problem (7) due to the less number of variables.

Next, we show that $\bar{F}_k$ is a DC function.

Proposition 3.2. *Let $\bar{F}_k$ be the function defined in* (9)*. Then $\bar{F}_k$ is a DC function and its DC decomposition can be written in the form*

$$\bar{F}_k(\boldsymbol{u}, v) = \bar{F}_k^1(\boldsymbol{u}, v) - \bar{F}_k^2(\boldsymbol{u}, v),$$

where the DC components are

$$\bar{F}_k^1(\boldsymbol{u}, v) = \sum_{i=1}^{m} \Big(r_{k-1}^i + \phi_i(\boldsymbol{u}, v) \Big) \quad \textit{and}$$
$$\bar{F}_k^2(\boldsymbol{u}, v) = \sum_{i=1}^{m} \max \Big\{ r_{k-1}^i,\, \phi_i(\boldsymbol{u}, v) \Big\}.$$

Proof. The DC decomposition is obtained by noticing that in (9) the term $\min\{r_{k-1}^i, \phi_i(\boldsymbol{u}, v)\}$ is a minimum of convex functions. Thus, we can apply the case (i) of Lemma 2.3. This directly yields the result. □

4. Double bundle method for CSVLR problems

In this section, we present a modified double bundle method (DBDC-CSVLR) to solve CSVLR problems. The main idea in the new DBDC-CSVLR method is to combine the best features of the double bundle method (DBDC) (Joki et al., 2018) and the incremental algorithm (Bagirov et al., 2013) using the CSVLR formulation. The DBDC is a local search method for nonsmooth DC optimization utilizing explicitly a DC decomposition of the objective function to take advantage of both the convexity and the concavity of the objective. This way the nonconvex cutting plane model represents the behaviour of the nonconvex objective better than a convex model. Moreover, the DBDC is applied to solve the CSVLR problem (7) and the auxiliary CSVLR problem (8) at each iteration of the incremental algorithm. A more detailed description of the algorithms follows.

4.1. DBDC

For simplicity, we describe the DBDC for a DC function f defined on the n-dimensional space $\mathbb{R}^n$. The DC representation of a function $f = f^1 - f^2$ has a central role in the method. We assume that at each point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate the values of the DC components $f^1(\boldsymbol{x})$ and $f^2(\boldsymbol{x})$ as well as arbitrary subgradients $\boldsymbol{\xi}^1 \in \partial f^1(\boldsymbol{x})$ and $\boldsymbol{\xi}^2 \in \partial f^2(\boldsymbol{x})$, respectively. Note that this assumption is trivially satisfied for both the CSVLR and the auxiliary CSVLR problems.

The main idea is to treat the DC components f^1 and f^2 separately in the model construction. Therefore, we also form separate approximations of the

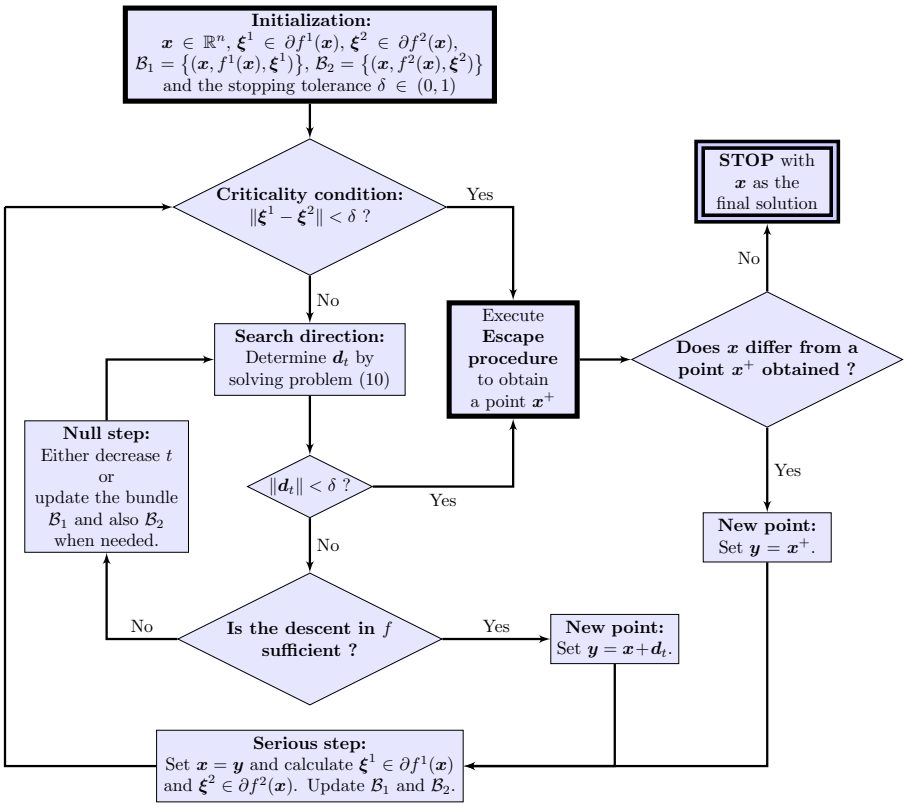


Figure 1: DBDC

subdifferentials of these components. This is done by collecting subgradient information from the previous iterations into two bundles, which are represented as

$$\mathcal{B}_i = \{(\boldsymbol{y}_j, f^i(\boldsymbol{y}_j), \boldsymbol{\xi}_j^i) \,|\, j \in J_i\} \quad \text{for } i = 1, 2,$$

where $\boldsymbol{y}_j \in \mathbb{R}^n$ is an auxiliary point, $\boldsymbol{\xi}_j^i \in \partial f^i(\boldsymbol{y}_j)$ and J_i is a nonempty set of indices. With this information we construct a *convex cutting plane model*

$$\hat{f}^i(\boldsymbol{x}) = \max_{j \in J_i} \left\{ f^i(\boldsymbol{y}_j) + (\boldsymbol{\xi}_j^i)^T (\boldsymbol{x} - \boldsymbol{y}_j) \right\}$$

for the DC component f^i, $i = 1, 2$. This model is the one used in convex bundle methods (Kiwiel, 1990; Mäkelä, 2002; Schramm and Zowe, 1992).

The overall approximation of f is obtained by combining the separate models

of the DC components. Thus, the *nonconvex cutting plane model* of f is

$$\hat{f}(\boldsymbol{x}) = \hat{f}^1(\boldsymbol{x}) - \hat{f}^2(\boldsymbol{x})$$

and, due to its structure, it takes into account both the convex and the concave behaviour of f.

To determine the search direction $\boldsymbol{d}_t \in \mathbb{R}^n$ at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$, we need to globally solve the following nonsmooth DC minimization problem

$$\begin{cases} \min & \hat{f}(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s.t.} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{10}$$

The quadratic term in this problem is a stabilizing term and the parameter $t > 0$ is the proximity measure used in most bundle methods. Due to the nonconvexity of the problem (10), the challenge is to find the global solution. However, the objective in this problem has a special DC structure and, thus, the global solution can be obtained quite easily by using an approach presented in Le Thi and Pham Dinh (1997, 2005) and utilized in Joki et al. (2017a).

When the direction $\boldsymbol{d}_t$ is found, characteristic to bundle methods is to decide whether to execute a *serious step* or a *null step*. In order to take a serious step, the following descent condition

$$f(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k) \leq \hat{m}\left(\hat{f}(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k)\right) \tag{11}$$

needs to be satisfied, where $\hat{m} \in (0, 1)$ is the descent parameter and $\hat{f}(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k) < 0$ is the predicted descent. This guarantees that the value of the objective f decreases sufficiently and, thus, we can update the iteration point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_t$. If the condition (11) does not hold, then the current model is not accurate enough and we need to execute a null step. In this step, the aim is to improve the model by adjusting the proximity measure t or updating the bundles and, therefore, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.

The sequence of serious and null steps is executed until a stopping criterion is satisfied. This requires that either the current iteration point $\boldsymbol{x}_k$ is critical

satisfying $\|\boldsymbol{\xi}^1 - \boldsymbol{\xi}^2\| < \delta$ or $\|\boldsymbol{d}_t\| < \delta$, where $\delta > 0$ is the stopping tolerance used in the algorithm. After finding such a point we execute the escape procedure (Joki et al., 2018) and it generates a new point $\boldsymbol{x}^+$. If $\boldsymbol{x}^+$ is the same as the current iteration point $\boldsymbol{x}_k$, then Clarke stationarity of the point $\boldsymbol{x}_k$ is ensured and the algorithm terminates. Otherwise, a solution candidate $\boldsymbol{x}_k$ is not Clarke stationary. In this case, the escape procedure generates a descent direction and we apply the DBDC starting from a better point $\boldsymbol{x}^+$.

The basic structure of the DBDC is presented in Figure 1. Suitable starting points for the algorithm are obtained by utilizing the incremental algorithm presented in the next subsection.

Before recalling the convergence results of the DBDC, we state the following assumptions:

A1 The set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is compact for a starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$.

A2 The subdifferentials $\partial f^1(\boldsymbol{x})$ and $\partial f^2(\boldsymbol{x})$ are polytopes at each $\boldsymbol{x} \in \mathbb{R}^n$.

These assumptions are trivially satisfied for both the CSVLR and the auxiliary CSVLR problems.

Lemma 4.1. *(Joki et al., 2018) Let the assumptions **A1** and **A2** be valid. During the DBDC, the execution of the escape procedure stops after a finite number of iterations.*

Theorem 4.2. *(Joki et al., 2018) Let the assumptions **A1** and **A2** be valid. For any $\tilde{\varepsilon} > 0$ and $\delta > 0$, the DBDC terminates after a finite number of iterations at a point $\boldsymbol{x}^*$ satisfying the approximate Clarke stationarity condition*

$$\|\boldsymbol{\xi}^*\| \leq \delta \quad \text{with } \boldsymbol{\xi}^* \in \partial^G_{\tilde{\varepsilon}} f(\boldsymbol{x}^*).$$

4.2. Incremental algorithm

Next, we introduce the incremental algorithm based on the method presented in Bagirov et al. (2013), but instead of CLR problems, it is modified to CSVLR problems. The incremental algorithm starts with the calculation of one hyperplane approximating the whole data and gradually adds one hyperplane

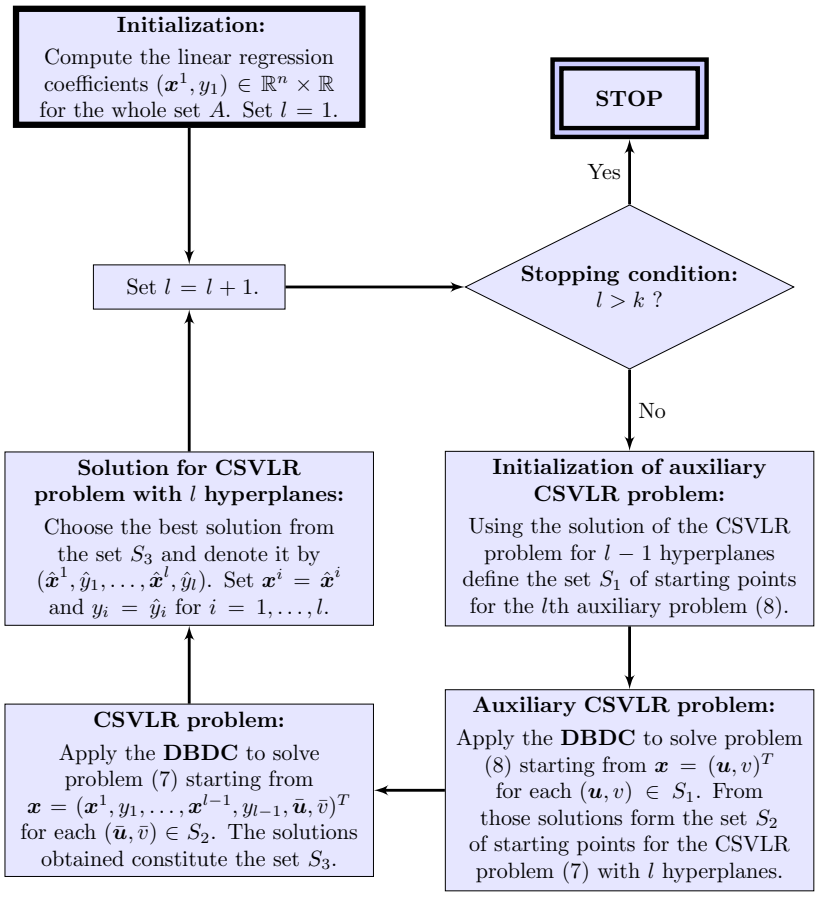


Figure 2: DBDC-CSVLR method combining the incremental algorithm and the DBDC

at each iteration until the required number of hyperplanes is calculated. During each iteration this method also utilizes the auxiliary problem (8) to generate a set of starting points for the CSVLR problem (7). Note that the auxiliary problem is much easier and less time-consuming to solve than (7).

Since the new DBDC-CSVLR method is based on the DBDC and the incremental algorithm it constructs clusters as well as linear functions approximating them incrementally. This means that the DBDC-CSVLR method do not solve just the CSVLR problem with k hyperplanes, but yields as the by-product solutions for each CSVLR problem with a smaller number of hyperplanes.

The detailed description of the DBDC-CSVLR method is presented in Figure 2, where S_1 denotes the set of starting points used at the initialization step for

solving the auxiliary CSVLR problem. This set is selected to be quite large whereas the set S_2, used to solve the CSVLR problem, is typically much smaller and contains only the best points obtained by using S_1. In addition, we denote by S_3 the set consisting of the solutions for the CSVLR problem (7). The detailed description of the sets S_1, S_2 and S_3 is given in Bagirov et al. (2013).

5. Numerical results

The aim of this section is threefold. First, using some synthetic data sets we study sensitivity of the CSVLR problem (7) to outliers. Since CLR is typically modelled utilizing the piecewise quadratic fit function we compare the results of the CSVLR problem with the ones obtained for the PQ-CLR problem (4). This helps us to have a better understanding about differences between the use of the L_1-riskand the L_2-risk in CLR. To solve the PQ-CLR model (4) we use the LMBM-CLR method (Karmitsa et al., 2016) which is a combination of the limited memory bundle method (Haarala et al., 2004, 2007) for large-scale nonsmooth optimization and the incremental algorithm (Bagirov et al., 2013). The second aim is to demonstrate the generalization ability of the proposed DBDC-CSVLR algorithm using some real-world data sets. Finally, we compare the DBDC-CSVLR algorithm with the multi-start exchange algorithm (MS-EA) (Späth, 1979), the expectation maximization algorithm (EM) (Dempster et al., 1977) and the DC-CLR algorithm (Bagirov and Taheri, 2017) using real-world data sets. All of these solvers utilize the L_1-riskto define regression errors in the CLR problem.

The codes `DBDC-CSVLR`, `LMBM-CLR`, `MS-EA` and `DC-CLR` are implemented in Fortran 95 and compiled by using `gfortran`, the GNU Fortran compiler. In addition, the subroutine `PLQDF1` (Lukšan, 1984) is used in `DBDC-CSVLR` to solve the problem (10). The code `EM` is implemented in R by using the flexmix package (Grün and Leisch, 2008). Tests are performed on an Intel® Core™ i5-2400 CPU (3.10GHz) running on Windows 10. Fortran source code of `DBDC-CSVLR` including `LMBM-CLR` can be downloaded from http://napsu.karmitsa.fi/svmclr/.

Note that the stopping tolerance parameter δ in the DBDC-CSVLR method depends on the number of variables. The smaller the number of variables is, the tighter is the value of this parameter. Since the auxiliary problem (8) has less

number of variables than the problem (7) we are able to solve it more accurately. Therefore, we select a smaller stopping tolerance for the problem (8) than for the problem (7). That is we select this parameter as

$$\delta = \begin{cases} 10^{-3}, & \text{for the CSVLR problem (7)} \\ 10^{-4}, & \text{for the auxiliary CSVLR problem (8).} \end{cases}$$

This selection also allows us to compute starting points which are close to the solutions of the problem (7). The size of $\mathcal{B}_1$ is set to 50. For $\mathcal{B}_2$ the size is *one* for the problem (8) and 3 otherwise. In the escape procedure, we use a bundle with the size 100. The descent parameter $\hat{m}$ is set to 0.2. For the other parameters, we apply the default values, but we set the value 10^{11} for the increase parameter (see Joki et al. (2018) for details).

To simplify the selection of ε during the execution of `DBDC-CSVLR`, we first normalize input and output variables in a data set so that they have zero mean value with the standard deviation one. Due to this, in normalized data ε can be seen as some sort of a "percent" coverage around a hyperplane meaning that, for example, $\varepsilon = 0.05$ covers roughly 5 % of the output values. Therefore, in the following the parameter ε is selected for the normalized data and not for the original one.

We use default values for parameters of `LMBM-CLR` and `DC-CLR` (Karmitsa et al., 2016; Bagirov and Taheri, 2017) and in `MS-EA` the simple randomized multistart scheme is applied to obtain starting points. Furthermore, in `EM` parameters are the default ones given in the flexmix package. In addition, we denote by k the number of hyperplanes (or clusters), by m the number of data points and by n the number of input variables.

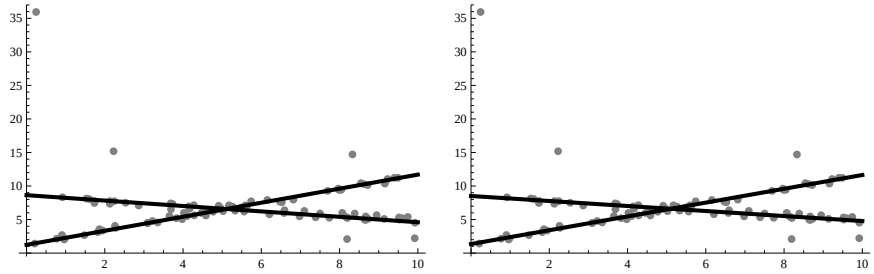
5.1. Model construction and effect of outliers

In this subsection, using six simple synthetic data sets, we analyse sensitivity of the CSVLR problem (7) to outliers and demonstrate the difference between the L_1-risk and the L_2-risk. The first three data sets are generated using known hyperplanes and the rest are generated clusterwise. In addition, all the data sets include some outliers and have one input and one output variable to allow visualization of results. The number of data points ranges from 100 to 1420.

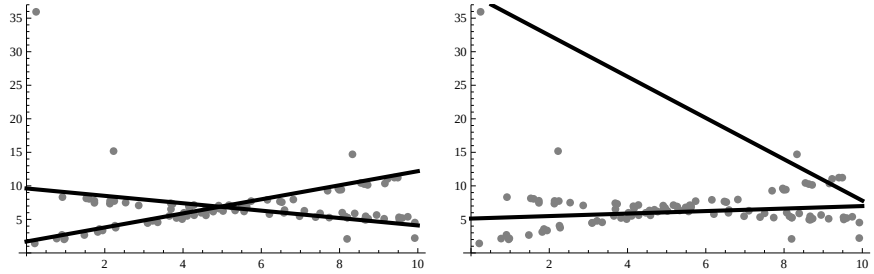
The description of the data sets is given in Table 1.

Table 1: The description of the synthetic data sets

Data set	m
Two Lines	100
Three Lines	999
Four Lines	1100
Clusters 1	190
Clusters 2	1420
Clusters 3	1320



a) The solution for $\varepsilon = 0.0$ b) The solution for $\varepsilon = 0.05$



c) The solution for $\varepsilon = 0.25$ d) The solution for $\varepsilon = 0.5$

Figure 3: Results for Two Lines data set with `DBDC-CSVLR`, $k = 2$ and different values of the parameter ε

Let us first discuss the impact of the value of the parameter ε on the performance of `DBDC-CSVLR`. Two lines data set set is used for this purpose. This data set is generated using two known lines by adding some random noise and two distinct outliers. The results are given in Figure 3. We can see that there is no difference between the solution for $\varepsilon = 0.05$ and that of for $\varepsilon = 0.0$. Moreover, in both cases obtained hyperplanes describe data correctly. However, when the parameter ε increases the influence of outliers increases as well. Therefore, the parameter ε should not be taken too large since this can completely distort the

solution as demonstrated in Figure 3 for $\varepsilon = 0.5$. The fact that the solution for $\varepsilon = 0.05$ is not affected by outliers justifies the use of small positive values for ε. Therefore, in the rest of this subsection we set $\varepsilon = 0.05$.

The solution for Two Lines data set with `LMBM-CLR` and $k = 2$ is presented in Figure 4. We can see that in this case the solution is considerably affected by outliers. This means that the use of the PQ-CLR model (4) does not lead to finding of correct hyperplanes since two outliers distort the solution. Thus, in this example the CSVLR model (7) is more robust to outliers than (4).

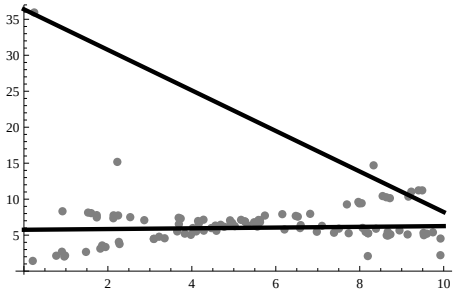
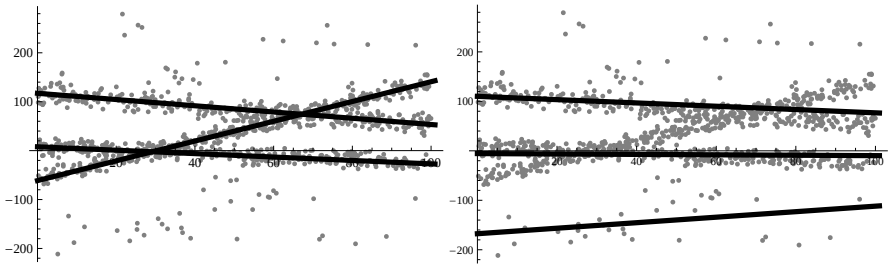


Figure 4: Result for Two Lines data set with `LMBM-CLR` and $k = 2$



a) The CSVLR model (7) b) The PQ-CLR model (4)

Figure 5: Results for Three Lines data set with $k = 3$

The results for Three Lines data set with both the new model (7) and the PQ-CLR model (4) are presented in Figure 5. This data set is generated using three different lines by adding some noise and outliers. The results show that `LMBM-CLR` based on the PQ-CLR model finds two out of three lines and the third line approximates the outliers. However, `DBDC-CSVLR` is able to distinguish correctly all the hyperplanes from the outliers. Thus, this example also confirms that the new model has ability to distinguish outliers.

Figure 6 illustrates the progress of `DBDC-CSVLR` in Four lines data set with

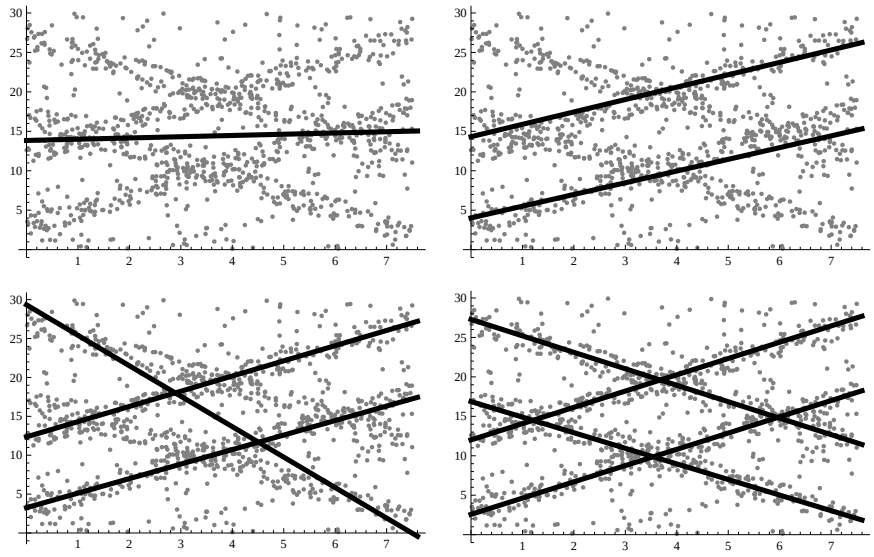


Figure 6: Progress of DBDC-CSVLR in Four Lines data set with $k = 4$

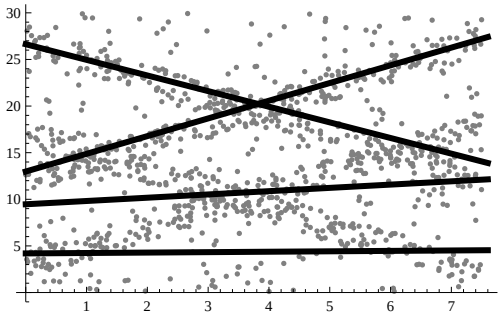


Figure 7: Result for Four Lines data set with the PQ-CLR model (4) and $k = 4$

$k = 4$. Note that we include all intermediate (or by-product) solutions ($k = 1, 2, 3$ and 4) obtained with DBDC-CSVLR for this data set. Similar to the previous example, this data set is generated using four known hyperplanes and adding some random noise and outliers. Results for this data set show that the number of hyperplanes should be chosen carefully since it affects the final result and how well it describes data. However, in this example even in the cases $k = 2$ and $k = 3$, the use of the CSVLR model allows to correctly identify two hyperplanes.

The solution for Four Lines data set with the PQ-CLR model (4) and $k = 4$ is presented in Figure 7. This example shows that the algorithm based on the PQ-CLR model, in general, fails when data points cover the space almost evenly. Here, two hyperplanes are placed quite correctly and they affect the positioning

of the other two hyperplanes, which are just placed to cover the area of data evenly.

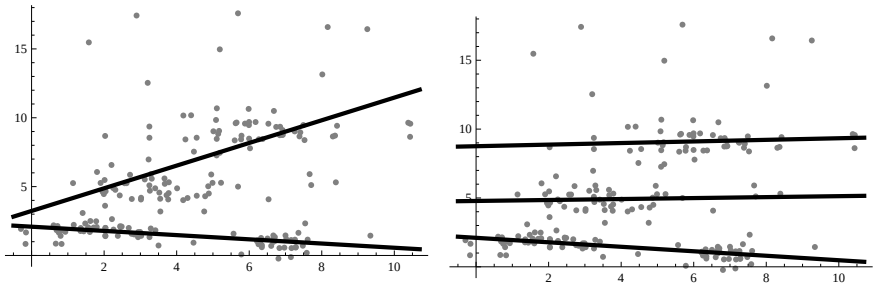


Figure 8: Progress of `DBDC-CSVLR` in Clusters 1 data set with $k = 3$

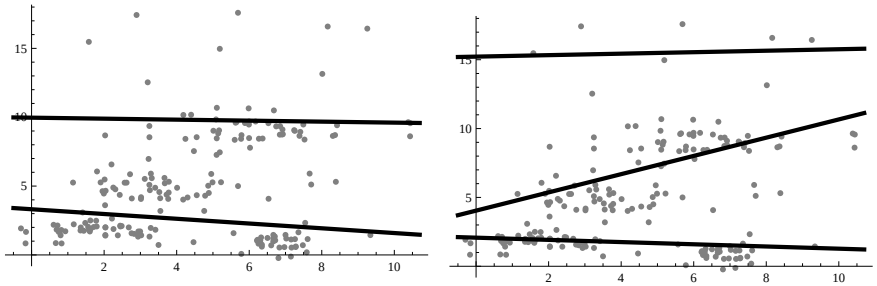
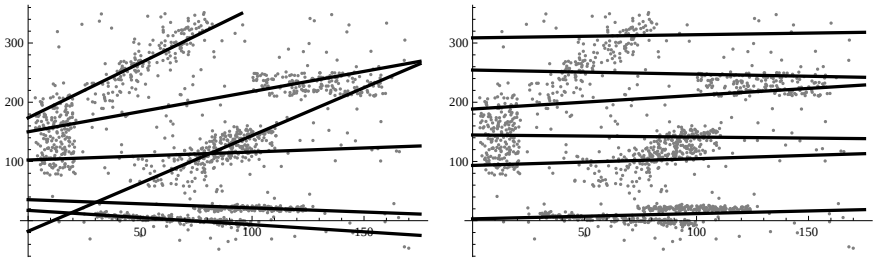


Figure 9: Progress of `LMBM-CLR` in Clusters 1 data set with $k = 3$



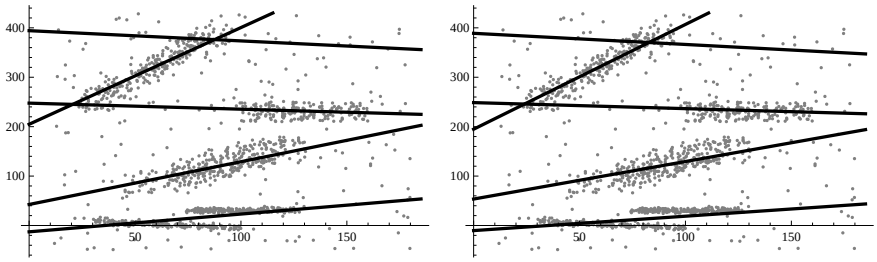
a) The CSVLR model (7) b) The PQ-CLR model (4)

Figure 10: Results for Clusters 2 data set with $k = 6$

Next, we consider Clusters 1 data set, where there are four clusters with some outliers. The solutions to the problems (7) and (4) are presented in Figures 8 and 9, respectively. In the case $k = 2$, `LMBM-CLR` produces an inaccurate solution since one hyperplane is positioned between clusters and fails to provide an accurate approximation of clusters. However, `DBDC-CSVLR` finds more accu-

rate solution in the case $k = 2$ since hyperplanes provide an approximation for two clusters. Moreover, in the case $k = 3$ the solution obtained by DBDC-CSVLR describes data almost correctly, whereas LMBM-CLR provides a solution where one hyperplane only approximates the outliers.

The results for the largest data set, Clusters 2 are presented in Figure 10. This data contains six clusters and the vertical cluster on the left is difficult to separate since it can be approximated with the hyperplanes of other clusters. In general, the vertical clusters are very difficult to be identified in regression problems since the slope of a hyperplane in regression cannot be infinity. DBDC-CSVLR is able to find correctly four out of six hyperplanes. Since the vertical cluster is covered with three different hyperplanes this affects the final solution and the algorithm fails to find an approximation for one horizontal cluster. Results for this data set show that LMBM-CLR based on the model (4) is sensitive to vertical clusters. Thus, the horizontal clusters are presented in the solution, but the structure of the other clusters is not captured.



a) The CSVLR model (7) b) The PQ-CLR model (4)

Figure 11: Results for Clusters 3 data set with $k = 5$

In Figure 11, we illustrate the solutions for Clusters 3 data set obtained by DBDC-CSVLR and LMBM-CLR. This data set resembles Clusters 2 data set, but the scale of the output is a little bit different and it does not contain a vertical cluster. The solutions obtained with DBDC-CSVLR and LMBM-CLR are nearly the same and approximate most of data accurately even though one hyperplane in both solutions only approximates the outliers. In addition, by comparing Figures 10 and 11 we notice that the absence of the vertical cluster can improve the solution considerably since both algorithms are not able to detect this kind of clusters. It is also worth noting that in DBDC-CSVLR by decreasing the value of

the parameter ε we are able to improve the solution, since the smaller ε makes it possible to separate the two narrow clusters close to each other at the bottom. For example, the solution obtained with $\varepsilon = 0.02$ is presented in Figure 12 and it detects correctly all clusters.

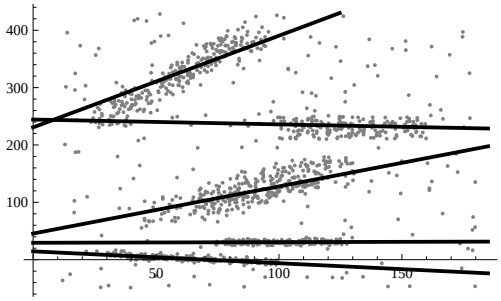


Figure 12: Result for Clusters 3 data set with `DBDC-CSVLR`, $k = 5$ and $\varepsilon = 0.02$

5.2. Generalization ability

In this subsection, we study the generalization ability of the DBDC-CSVLR method using some real-world data sets. These data sets are divided into two parts: a training set and a test set. These sets are chosen randomly such a way that 90 % of points in a data set forms the training set and the rest 10 % of them forms the test set. We repeat this procedure ten times for each data set, that is we use the ten-fold cross validation.

The brief description of the real-world data sets is presented in Table 2. The detailed description can be found in Dua and Graff (2019) and the references given in Table 2. All the data sets have only numeric input variables, whose numbers range from 4 to 11, and one numeric output variable. The number of data points in these data sets ranges from 1030 to 45730. In addition, the data sets contain no missing values.

In each data set, hyperplanes are first found by the DBDC-CSVLR algorithm using the training set with the selection $k = 10$ ($k = 6$ in Red wine quality data set) whereas the results for the smaller k are the by-products obtained from the incremental algorithm. Once hyperplanes are obtained they are tested on the test set. In order to estimate the performance of this algorithm, we use the

Table 2: The brief description of the real-world data sets

Data set	m	n	Reference
Concrete compressive strength	1030	8	Yeh (1998)
Airfoil self-noise	1503	5	Dua and Graff (2019)
Red wine quality	1599	11	Cortez et al. (2009)
Combined cycle power plant	9568	4	Kaya et al. (2012); Tüfekci (2014)
Physicochemical properties of protein tertiary structure	45730	9	Dua and Graff (2019)

Root Mean Square Error (RMSE)

$$\mathrm{RMSE} = \left(\frac{1}{|I|} \sum_{i \in I} (\bar{b}_i - b_i)^2 \right)^{1/2},$$

where I is the index set of a training or a test set, $|I|$ is the cardinality of I, b_i is the actual observed output value for a point with $i \in I$ and $\bar{b}_i$ is the estimated output value for a point with $i \in I$ obtained by using the closest hyperplane of the adjusted ones. We calculate RMSE for both training and test sets. Since we apply the ten-fold cross validation we report the average RMSE values in Tables 3–7. The closer the RMSE values of training and test sets are, the better the generalization ability is. Due to the normalization of data, we use for the parameter ε the values 0.0, 0.01 and 0.05 in the normalized data sets. However, this does not affect the obtained RMSE values since they do not depend on ε.

Furthermore, in Tables 3–7, we report the CPU time required by the DBDC-CSVLR method in the training phase. Due to the ten-fold cross validation, CPU is the average computation time in seconds over k. Note also that the difference in CPU between two consecutive k values is how much more time is required in average to adjust one more hyperplane to the solution. Thus CPU time, for example, for $k = 10$ contains also the computational time to solve each smaller CSVLR problem with $k = 1, \ldots, 9$, since they are obtained as a by-product with the DBDC-CSVLR algorithm.

From Tables 3–7 we see that for each ε the average RMSE values for training and test sets are quite similar meaning that the DBDC-CSVLR method has a good generalization ability. The only exception is Concrete compressive strength data set where the differences are slightly larger. In addition, we notice that the

Table 3: Average RMSE values for Concrete compressive strength data set

	$\varepsilon = 0.0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
k	Training	Test	CPU	Training	Test	CPU	Training	Test	CPU
1	10.7509	12.3205	0.04	10.7424	12.3382	0.04	10.6951	12.1051	0.04
2	5.7553	6.6335	10.71	5.7943	6.6790	13.04	5.7813	6.6312	8.57
3	3.9375	5.1470	37.94	3.9519	5.1769	37.56	3.8876	4.6621	27.70
4	3.0352	4.3873	81.39	3.0357	3.9117	70.75	2.9851	3.7071	49.83
5	2.5152	3.8583	119.97	2.5467	3.4633	105.61	2.4792	3.2845	62.04
6	2.1175	3.3825	184.95	2.1097	3.2133	147.19	2.0174	2.9492	69.04
7	1.8118	3.0971	258.53	1.7931	3.0047	207.41	1.7223	2.7115	82.90
8	1.5221	2.9675	342.50	1.5380	2.8168	258.68	1.4789	2.4923	97.94
9	1.3403	2.8294	428.05	1.3259	2.6374	314.11	1.2998	2.3410	117.09
10	1.1825	2.5772	555.13	1.1826	2.4921	370.32	1.1550	2.1488	150.99

Table 4: Average RMSE values for Airfoil self-noise data set

	$\varepsilon = 0.0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
k	Training	Test	CPU	Training	Test	CPU	Training	Test	CPU
1	4.8680	4.9371	0.02	4.8674	4.9345	0.02	4.8299	4.8831	0.03
2	2.8704	3.2597	11.93	2.8712	3.2241	13.68	2.8102	3.1541	4.50
3	2.0372	2.4475	22.57	2.0312	2.3682	29.67	2.0238	2.3862	21.29
4	1.6433	1.9784	42.65	1.6467	1.9369	51.24	1.6115	1.9780	42.74
5	1.3575	1.7148	70.88	1.3469	1.6940	78.66	1.3537	1.7709	53.97
6	1.1843	1.5838	103.81	1.1538	1.5351	114.08	1.2060	1.4662	59.70
7	1.0196	1.4701	148.67	0.9975	1.3547	132.32	1.1005	1.3204	67.01
8	0.8804	1.2777	195.59	0.8630	1.2330	157.77	1.0305	1.1932	69.44
9	0.8040	1.1975	243.42	0.7800	1.1806	174.68	0.9531	1.1021	72.75
10	0.7141	1.1650	303.40	0.6894	1.1537	199.98	0.9002	1.0579	73.94

Table 5: Average RMSE values for Red wine quality data set

	$\varepsilon = 0.0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
k	Training	Test	CPU	Training	Test	CPU	Training	Test	CPU
1	0.6523	0.6640	0.08	0.6523	0.6643	0.10	0.6491	0.6638	0.10
2	0.4768	0.4665	21.43	0.4531	0.4630	33.48	0.4029	0.4323	14.61
3	0.2633	0.2565	21.49	0.2588	0.2522	45.16	0.2611	0.2676	32.19
4	0.1322	0.1229	21.63	0.1287	0.1314	45.40	0.1504	0.1601	35.57
5	0.0789	0.0600	21.87	0.0764	0.0748	46.71	0.1123	0.1192	37.97
6	0.0000	0.0000	22.03	0.0173	0.0211	50.60	0.0904	0.0980	40.81

use of the small positive values of the parameter ε gives almost always smaller RMSE values than $\varepsilon = 0.0$. Furthermore, in most cases a larger value of ε often reduces the computational time. Note also that Red wine quality data set can be exactly represented by six linear functions. This may be one reason why any positive value of ε increases the computational time. All in all, the results clearly demonstrate that the use of the CSVLR model leads to algorithms with

Table 6: Average RMSE values for Combined cycle power plant data set

	$\varepsilon = 0.0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
k	Training	Test	CPU	Training	Test	CPU	Training	Test	CPU
1	4.5733	4.5735	0.19	4.5687	4.5687	0.20	4.5837	4.5844	0.19
2	2.7699	2.7686	21.28	2.7504	2.7492	12.44	4.0139	4.0180	2.36
3	2.1510	2.1471	51.82	2.0906	2.0846	23.44	3.8703	3.9152	10.57
4	1.7628	1.7538	103.95	1.7080	1.7036	157.51	3.6779	3.7081	24.25
5	1.5565	1.5403	225.80	1.4878	1.4418	217.65	3.6078	3.6368	26.13
6	1.4263	1.4075	310.93	1.2567	1.2787	284.65	3.1809	3.2152	27.45
7	1.3392	1.3192	400.54	1.0756	1.1278	353.45	2.8871	2.9136	30.86
8	1.2270	1.2041	540.79	0.9789	1.0233	394.02	2.3004	2.3189	34.60
9	1.1365	1.1071	713.80	0.9073	0.9606	448.18	1.8068	1.8383	38.59
10	1.0959	1.0659	898.28	0.8523	0.9174	493.30	1.6957	1.7294	42.90

Table 7: Average RMSE values for Physicochemical properties of protein tertiary structure data set

	$\varepsilon = 0.0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
k	Training	Test	CPU	Training	Test	CPU	Training	Test	CPU
1	5.3033	5.3068	6.14	5.3031	5.3067	6.18	5.3017	5.3050	6.32
2	2.2095	2.2105	57.22	2.2094	2.2103	61.13	2.2051	2.2062	62.79
3	1.5724	1.5783	265.96	1.5714	1.5794	275.07	1.5641	1.5698	288.70
4	1.2128	1.2189	517.15	1.2134	1.2171	536.73	1.2073	1.2114	561.32
5	0.9819	0.9866	852.15	0.9825	0.9870	880.92	0.9749	0.9806	910.95
6	0.8655	0.8705	1269.34	0.8653	0.8702	1304.49	0.8372	0.8431	1325.47
7	0.7214	0.7263	1755.65	0.7190	0.7234	1811.24	0.7144	0.7179	1861.81
8	0.6584	0.6628	2371.80	0.6558	0.6587	2422.92	0.6285	0.6315	2467.94
9	0.5867	0.5916	3037.26	0.5783	0.5831	3130.61	0.5685	0.5730	3223.65
10	0.5315	0.5371	3791.48	0.5279	0.5339	3904.30	0.5121	0.5163	4035.21

a good generalization ability when the parameter $\varepsilon > 0$ is small (e.g., 0.01 or 0.05). Larger values of ε may reduce the computational time even more, but at the same time lead to the loss of accuracy.

5.3. Comparison with other solvers

In this subsection, `DBDC-CSVLR` is compared with `MS-EA`, `EM` and `DC-CLR`. All of these algorithms solve CLR problems where the regression error is defined using the L_1-risk. However, the objective functions in these problems are different and, therefore, we cannot directly compare obtained solutions using the objective function values. Instead we use three performance measures to compare different solvers.

The comparisons are done in the real-world data sets presented in Table 2. To compare the results we use, for example, the Nash-Sutcliffe Coefficient of Efficiency (CE) and the Pearson correlation coefficient (r) calculated with the

formulas

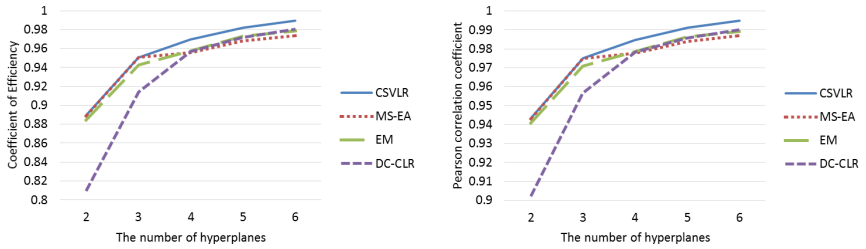
$$\mathrm{CE} = 1 - \left(\frac{\sum_{i=1}^{m} (b_i - \bar{b}_i)^2}{\sum_{i=1}^{m} (b_i - b_0)^2} \right) \quad \text{and}$$
$$r = \frac{\sum_{i=1}^{m} (b_i - b_0)(\bar{b}_i - \bar{b}_0)}{\sqrt{\sum_{i=1}^{m} (b_i - b_0)^2} \sqrt{\sum_{i=1}^{m} (\bar{b}_i - \bar{b}_0)^2}},$$

where b_i is the observed output value and $\bar{b}_i$ is the estimated output value obtained by using the closest hyperplane of the adjusted ones whereas b_0 and $\bar{b}_0$ are the means of the observed and the estimated output values, respectively. The measure CE ranges from $-\infty$ to 1 and the value $\mathrm{CE} = 1$ means a perfect prediction. On the other hand, the measure r ranges from -1 to 1 and a linear equation describes the relationship between observed and estimated values perfectly if $r = 1$. Therefore, the closer these measures are to one, the better the result of a method is.

In addition, we use the well-known adjusted Rand index (ARI) (Hubert and Arabie, 1985) to compare the CLR results produced by different algorithms. Since the true clusters (or regression functions) of the real-world data are not known, we use the results of the proposed algorithm DBDC-CSVLR as a benchmark. Note that the aim here is not to judge which method gives the best or correct clustering (for that we should know the correct clustering) but to compare the results obtained with different algorithms.

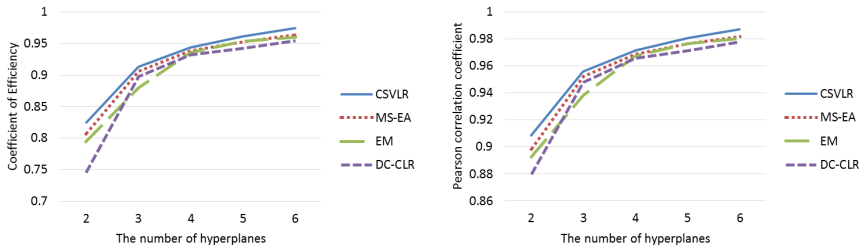
Results for the CE and r values are shown in Figures 13–17, where we present separately for each data set the obtained values of the measures CE and r when the number of the hyperplanes is increased. To highlight the differences between the solvers, we have not included the results with one hyperplane ($k = 1$), since in this case all the solvers yielded nearly the same values for CE and r in each data set. From Figures 13–17 we can conclude that the solver `DBDC-CSVLR` has the best performance in each data set even though `MS-EA` behaves often quite similarly.

We used the function `rand_index` in McComb (2020) to compute ARI. Table 8 shows the values of ARI with different number of clusters.The value of ARI equal to 1 indicates the



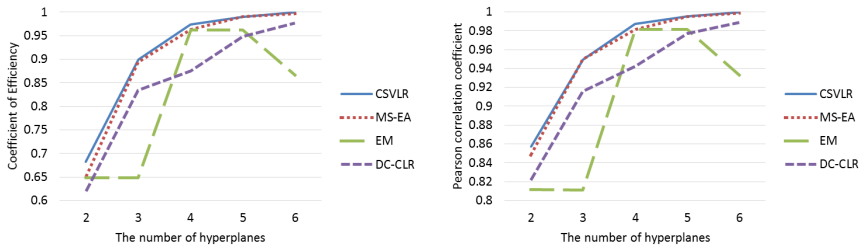
a) The measure CE b) The measure r

Figure 13: Results in Concrete compressive strength data set



a) The measure CE b) The measure r

Figure 14: Results in Airfoil self-noise data set

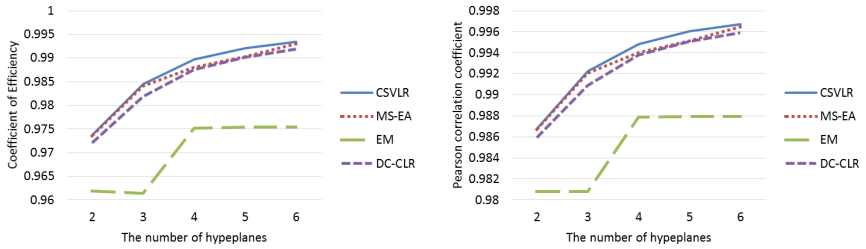


a) The measure CE b) The measure r

Figure 15: Results in Red wine quality data set

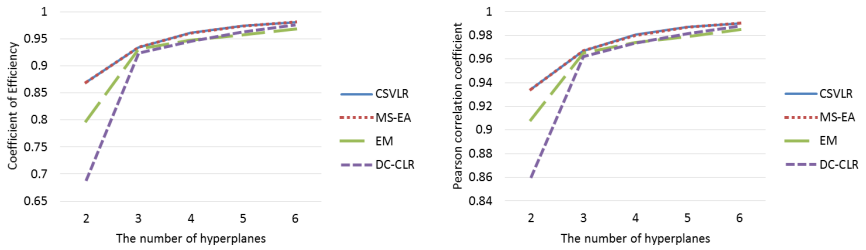
6. Conclusions

In this paper, we have introduced a new clusterwise support vector linear regression (CSVLR) model of the clusterwise linear regression (CLR) problem. The novelty in the formulation is that the support vector machine (SVM) approach is incorporated into the CLR problem. The new model enables us to tolerate perturbations from hyperplanes if they are smaller than the user specified tolerance $\varepsilon > 0$. The CSVLR problem is presented as a difference of



a) The measure CE b) The measure r

Figure 16: Results in Combined cycle power plant data set



a) The measure CE b) The measure r

Figure 17: Results in Physicochemical properties of protein tertiary structure data set

Table 8: Adjusted Rand indices in real-world data.

	Concrete compressive strength				Airfoil self-noise			
k	DBDC-CSVLR	MS-EA	EM	DC-CLR	DBDC-CSVLR	MS-EA	EM	DC-CLR
2	1.0000	0.9349	0.4772	0.0555	1.0000	0.9894	0.1309	0.1644
3	1.0000	0.2283	0.1766	0.1450	1.0000	0.9796	0.0628	0.4796
4	1.0000	0.4667	0.2213	0.1866	1.0000	0.3122	0.0914	0.2986
5	1.0000	0.1503	0.1168	0.1453	1.0000	0.2090	0.1468	0.3580
6	1.0000	0.1126	0.0730	0.1327	1.0000	0.1507	0.0614	0.2674
	Red wine quality				Combined cycle power plant			
k	DBDC-CSVLR	MS-EA	EM	DC-CLR	DBDC-CSVLR	MS-EA	EM	DC-CLR
2	1.0000	1.0000	0.0437	0.3083	1.0000	0.9842	0.0040	0.7715
3	1.0000	1.0000	0.0538	0.6727	1.0000	0.8646	0.0277	0.6367
4	1.0000	1.0000	0.9734	0.9235	1.0000	0.5577	0.1076	0.4601
5	1.0000	0.9976	0.9799	0.9723	1.0000	0.5167	0.1055	0.3928
6	1.0000	0.9995	0.9748	0.9833	1.0000	0.4586	0.1512	0.3211
	Protein tertiary structure							
k	DBDC-CSVLR	MS-EA	EM	DC-CLR				
2	1.0000	0.9976	0.8052	0.6627				
3	1.0000	0.9057	0.8876	0.6997				
4	1.0000	0.8874	0.6004	0.6336				
5	1.0000	0.8859	0.4801	0.5238				
6	1.0000	0.7473	0.5390	0.4270				

convex (DC) functions in order to capture both the convexity and the concavity of the objective function. In addition, we have proposed a new DBDC-CSVLR algorithm to solve the CSVLR problem. The proposed algorithm combines the double bundle method (DBDC) for nonsmooth DC optimization with the incremental algorithm. This combination enables us to find high quality solutions since they are constructed incrementally by solving the CSVLR problems and the auxiliary CSVLR problems with the DBDC method. In addition, by solving the CSVLR problem with the DBDC-CSVLR algorithm for k hyperplanes we obtain the by-product solutions for CSVLR problem with all smaller number of hyperplanes.

To validate the usefulness of the CSVLR formulation, we have tested the DBDC-CSVLR algorithm using six synthetic data sets. Three of these data sets are generated using known linear functions and three others using known clusters. Such a choice of data sets allows us to demonstrate the ability of the proposed algorithm in identifying linear functions for data sets with different structures. Numerical results show that the DBDC-CSVLR algorithm is efficient and robust to solve the CLR problems since in most cases it finds solutions describing data accurately. The comparison with the frequently used piecewise quadratic fit function also supports this conclusion as the CSVLR model outperforms the model based on this fit function in most synthetic data sets used in numerical experiments. In addition, the CSVLR model is able to tolerate outliers. This is an important feature since outliers are common in most real-world data sets and they can easily distort the solution.

We have also studied the generalization ability of the DBDC-CSVLR algorithm using five real-world data sets, three values of the tolerance ε for perturbations from hyperplanes and the performance measure RMSE. The results demonstrate that for all three values of ε the RMSE measures are similar in both training and test sets except in one data set. Furthermore, the performance of the proposed algorithm with a small $\varepsilon > 0$ is better than when $\varepsilon = 0.0$ which justifies the use of the SVM formulation of the CLR problem.

In addition, the new DBDC-CSVLR algorithm has been compared with three other regression methods utilizing L_1-riskto solve CLR problems. The comparisons have been performed in five real-world data sets and the results show that

in each data set the DBDC-CSVLR algorithm has the best performance among all solvers.

To conclude, the numerical results show that the new DBDC-CSVLR algorithm has a good performance and the generalization ability. Nevertheless, there are still some problems and open questions. The proposed algorithm is not able to identify the so-called "vertical" clusters. Second, it is not clear how efficient the DBDC-CSVLR algorithm is as a prediction tool. Moreover, the applicability of this algorithm for solving large-scale CLR problems or for supervised classification is not considered. These will be the subjects of future research.

Acknowledgments

We thank anonymous referees for their valuable comments and suggestions, which helped us to improve the paper. This work was funded by the Matti Programme of the University of Turku Graduate School UTUGS, the University of Turku, the Academy of Finland (Project No. 289500, 294002 and 319274) and the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580).

References

Bagirov, A.M., Karmitsa, N. & Mäkelä, M.M. (2014). *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, Cham, Heidelberg.

Bagirov, A.M., Mahmood, A. & Barton, A. (2017). Prediction of monthly rainfall in Victoria, Australia: Clusterwise linear regression approach. *Atmospheric Research*, *188*(15), 20–29.

Bagirov, A.M. and Taheri, S. (2017). DC programming algorithm for clusterwise linear L1 regression. *Journal of the Operations Research Society of China*, *5*(2), 233–256.

Bagirov, A.M. and Ugon, J. (2018). Nonsmooth DC programming approach to clusterwise linear regression: optimality conditions and algorithms. *Optimization Methods and Software*, *33*(1), 194–219.

Bagirov, A.M., Ugon, J. & Mirzayeva, H. (2013). Nonsmooth nonconvex optimization approach to clusterwise linear regression problems. *European Journal of Operational Research*, *229*(1), 132–142.

Bagirov, A.M., Ugon, J. & Mirzayeva, H. (2015a). An algorithm for clusterwise linear regression based on smoothing techniques. *Optimization Letters*, *9*(2), 375–390.

Bagirov, A.M., Ugon, J. & Mirzayeva, H. (2015b). Nonsmooth optimization algorithm for solving clusterwise linear regression problems. *Journal of Optimization Theory and Applications*, *164*(3), 755–780.

Bertsimas, D. & Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, *55*(2), 252–271.

Carbonneau, R.A., Caporossi, G. & Hansen, P. (2011). Globally optimal clusterwise regression by mixed logical-quadratic programming. *European Journal of Operational Research*, *212*(1), 213–222.

Carbonneau, R.A., Caporossi, G. & Hansen, P. (2012). Extensions to the repetitive branch and bound algorithm for globally optimal clusterwise regression. *Computers & Operations Research*, *39*(11), 2748–2762.

Clarke, F.H. (1983). *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York.

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

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, *47*(4), 547–553. Data set available in UCI machine learning repository `http://archive.ics.uci.edu/ml` (June 11th, 2016)

Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, *39*(1), 1–38.

DeSarbo, W.S. & Cron, W.L. (1988). A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, *5*(2), 249–282.

DeSarbo, W.S., Oliver, R.L. & Rangaswamy, A. (1989). A simulated annealing methodology for clusterwise linear regression. *Psychometrika*, *54*(4), 707–736.

Dua, D. & Graff, C. (2019). UCI machine learning repository. Available in web page `http://archive.ics.uci.edu/ml`, University of California, Irvine, School of Information and Computer Sciences. (April 8th, 2016)

D'Urso, P., Massari, R. & Santoro, A. (2010). A class of fuzzy clusterwise regression models. *Information Sciences*, *180*(24), 4737–4762.

Gaffney, S. & Smyth, P. (1999). Trajectory clustering using mixtures of regression models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 63–72.

Ganjigatti, J., Pratihar, D.K. & Choudhury, A.R. (2007). Global versus cluster-wise regression analyses for prediction of bead geometry in MIG welding process. *Journal of Materials Processing Technology*, *189*(1–3), 352–366.

Garcìa-Escudero, L.A., Gordaliza, A., Mayo-Iscar, A. & San Martin, R. (2010). Robust clusterwise linear regression through trimming. *Computational Statistics & Data Analysis*, *54*(12), 3057–3069.

Grün, B. & Leisch, F. (2008). FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, *28*(4), 1–35.

Haarala, M., Miettinen, K. & Mäkelä, M.M. (2004). New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software*, *19*(6), 673–692.

Haarala, N., Miettinen, K. & Mäkelä, M.M. (2007). Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, *109*(1), 181–205.

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

Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of Classification*,*2*, 193–218.

Joki, K., Bagirov, A.M., Karmitsa, N. & Mäkelä, M.M. (2017a). A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. *Journal of Global Optimization*, *68*(3), 501–535.

Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M. & Taheri, S. (2017b). New bundle method for clusterwise linear regression utilizing support vector machines. TUCS Technical Report No 1190, Turku Centre for Computer Science, Turku.

Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M. & Taheri, S. (2018). Double bundle method for finding Clarke stationary points in nonsmooth DC programming. *SIAM Journal on Optimization*, *28*(2), 1892–1919.

Karmitsa, N., Bagirov, A.M. & Taheri, S. (2016). Limited memory bundle method for solving large clusterwise linear regression problems. TUCS Technical Report No 1172, Turku Centre for Computer Science, Turku.

Kaya, H., Tüfekci, P. & Gürgen, S.F. (2012). Local and global learning methods for predicting power of a combined gas & steam turbine. Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012, pp. 13–18, March, Dubai. Data set available in UCI machine learning repository `http://archive.ics.uci.edu/ml` (June 11th, 2016)

Kiwiel, K.C. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, *46*(1–3), 105–122.

Lau, K., Leung, P. & Tse, K. (1999). A mathematical programming approach to clusterwise regression model and its extensions. *European Journal of Operational Research*, *116*(3), 640–652.

Le Thi, H.A. & Pham Dinh, T. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization*, *11*(3), 253–285.

Le Thi, H.A. & Pham Dinh, T. (2005). The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, *133*(1–4), 23–46.

Lukšan, L. (1984). Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minmax approximation. Kybernetika, *20*(6), 445–457.

McComb, C. (2020). Adjusted Rand Index. GitHub.`https://www.github.com/cmccomb/rand_index` Accessed 23 December 2019.

Mäkelä, M.M. (2002). Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, *17*(1), 1–29.

Mäkelä, M.M. & Neittaanmäki, P. (1992). *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., Singapore.

Park, Y.W., Jiang, Y., Klabjan, D. & Williams, L. (2017). Algorithms for generalized clusterwise linear regression. *INFORMS Journal on Computing*, *29*(2), 301–317.

Pham Dinh, T. & Le Thi, H.A. (1997). Convex analysis approach to D.C. programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, *22*(1), 289–355.

Poggi, J.-M. & Portier, B. (2011). PM10 forecasting using clusterwise regression. *Atmospheric Environment*, *45*(38), 7005–7014.

Preda, C. & Saporta, G. (2005). Clusterwise PLS regression on a stochastic process. *Computational Statistics & Data Analysis*, *49*(1), 99–108.

Rockafellar, R.T. (1970). *Convex Analysis*. Princeton University Press, Princeton, New Jersey.

Schramm, H. & Zowe, J. (1992). A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, *2*(1), 121–152.

Smola, A. J. & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Späth, H. (1979). Algorithm 39: Clusterwise linear regression. *Computing*, *22*(4), 367–373.

Späth, H. (1986). Clusterwise linear least absolute deviations regression. *Computing*, *37*(4), 371–378.

Strekalovsky, A.S. (2015). On local search in d.c. optimization problems. *Applied Mathematics and Computation*, *255*, 73–83.

Tuy, H. (1998). *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrecht.

Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, *60*, 126–140. Data set available in UCI machine learning repository `http://archive.ics.uci.edu/ml` (June 11th, 2016)

Wedel, M. & Kistemaker, C. (1989). Consumer benefit segmentation using clusterwise linear regression. *International Journal of Research in Marketing*, *6*(1), 45–59.

Yeh, I. (1998). Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, *28*(12), 1797–1808. Data set available in UCI machine learning repository `http://archive.ics.uci.edu/ml` (June 11th, 2016)