Kaisa Joki | Adil M. Bagirov | Napsu Karmitsa | Marko M. Mäkelä | Sona Taheri

New bundle method for clusterwise linear regression utilizing support vector machines

Turku Centre *for* Computer Science

TUCS Technical Report
No 1190 , December 2017

New bundle method for clusterwise linear regression utilizing support vector machines

Kaisa Joki
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
kaisa.joki@utu.fi

Adil M. Bagirov
Faculty of Science and Technology, Federation University Australia
University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia
a.bagirov@federation.edu.au

Napsu Karmitsa
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
napsu@karmitsa.fi

Marko M. Mäkelä
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
makela@utu.fi

Sona Taheri
Faculty of Science and Technology, Federation University Australia
University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia
s.taheri@federation.edu.au

TUCS Technical Report

No 1190 , December 2017

Abstract

Clusterwise linear regression (CLR) aims to simultaneously partition a data into a given number of clusters and find regression coefficients for each cluster. In this paper, we propose a novel approach to 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 CLR is represented as an unconstrained nonsmooth optimization problem, where the objective function is a difference of convex (DC) functions. A method based on the combination of the incremental algorithm and the double bundle method for DC optimization is designed to solve it. Numerical experiments are made to validate the reliability of the new formulation and the efficiency of the proposed method. The results show that the SVM approach is beneficial in solving CLR problems, especially, when there are outliers in data.

Keywords: Nonsmooth optimization, Support vector machines, Clusterwise regression, Incremental algorithm, Bundle method, DC functions

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Clusterwise linear regression (CLR) is a technique for fitting multiple hyperplanes to mutually exclusive subsets of observations of a data set [32]. It is a combination of two techniques: clustering and regression. Applications of CLR include, for example, the consumer benefit segmentation [36], market segmentation stock-exchange [28], metal inert gas welding process [12], rainfall prediction [2] and PM10 prediction [27]. To date, different models of the CLR problem have been proposed and algorithms have been developed for them. These algorithms include those based on extensions of k-means [32] and expectation-maximization (EM) [11] as well as on mixed integer nonlinear programming [6, 10], nonsmooth optimization [3, 4, 5] and mixture models [9, 13].

In this paper, a new approach for solving CLR problems is proposed using the support vector machines (SVM) for the regression method [8, 31]. By applying the SVM formulation for regression, the CLR problem is expressed as a constrained nonsmooth optimization problem. Then using the penalty function this problem is replaced by an unconstrained nonsmooth optimization problem. The objective function in the latter problem is represented as a difference of convex (DC) functions allowing us to use some results from convex analysis and optimization. To utilize the DC structure, the proposed algorithm uses the double bundle method (DBDC) [18] developed for nonsmooth DC optimization. However, the DBDC is a local method. Therefore, to obtain global or near global solutions, the DBDC is combined with an incremental approach introduced in [3] to design a more accurate algorithm for solving the nonconvex CLR problem. The algorithm is tested using some data sets for regression to validate the usability 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 DB-SVM-CLR combining the double bundle method DBDC and the incremental algorithm to solve CLR problems. 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 [1, 7, 16, 22, 26, 33, 35].

We denote by $\mathbb{R}^n$ the n-dimensional Euclidean space. The inner product is denoted by $\boldsymbol{u}^T\boldsymbol{v} = \sum_{i=1}^{n} u_i v_i$ and the associated norm by $\|\boldsymbol{u}\| = (\boldsymbol{u}^T\boldsymbol{u})^{1/2}, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$. The set $B(\boldsymbol{x};\varepsilon) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon\}$ 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 [29]

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

being a nonempty, convex and compact set. For convex functions, we have some useful subdifferential calculus rules. The following lemma presents two of them.

Lemma 2.1. *[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

$$\partial g_c(\boldsymbol{x}) = \sum_{i=1}^k \partial f_c^i(\boldsymbol{x}).$$

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

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

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* [7] at a point $\boldsymbol{x} \in \mathbb{R}^n$ with respect to a direction $\boldsymbol{d} \in \mathbb{R}^n$ is

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

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

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \ f^0(\boldsymbol{x}; \boldsymbol{d}) \geq \boldsymbol{\xi}^T \boldsymbol{d} \quad \text{for all } \boldsymbol{d} \in \mathbb{R}^n \right\}.$$

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 [7], 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 [25]

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

This subdifferential is a generalization of $\partial f(\boldsymbol{x})$ since $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for each $\varepsilon \geq 0$ 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 a *DC function* if there exist two convex fuctions $f^1, f^2 : \mathbb{R}^n \to \mathbb{R}$ such that

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

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. One benefit of DC functions is that they preserve the DC structure under some simple operations frequently used in optimization as the following lemma demonstrates.

Lemma 2.3. *[35] Let $f_i = f_i^1 - f_i^2$ for $i = 1, \dots, k$ be DC functions. Then*

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

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

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

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

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 problem (1), it is necessary that $\partial f^2(\boldsymbol{x}^*) \subseteq \partial f^1(\boldsymbol{x}^*)$ [34]. 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 weaker necessary conditions are used:

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

and

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

It is known that any Clarke stationary point is also critical. However, the opposite claim is not always true.

3 SVM based clusterwise linear regression

In this section, we introduce a new formulation of the CLR problem. The idea is to apply the SVM approach to model this problem. This differs from the SVM for general regression problems since instead of regression with only one regression function we consider the SVM for CLR problems, where several regression functions are fitted. In addition, we give two different DC decompositions for the SVM-CLR formulation. Moreover, we introduce the auxiliary SVM-CLR problem used to find promising starting points for the original SVM-CLR problem. The SVM for general regression problems with one linear function is discussed in [8, 31].

Suppose that we are given a finite data set $A = \{(\boldsymbol{a}^i, b_i) \in \mathbb{R}^n \times \mathbb{R} \mid i = 1, \dots, m\}$. 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. In what follows, let A^j for $j = 1, \dots, k$ be *clusters* such that

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

and $\{\boldsymbol{x}^j, y_j\}$ be the corresponding linear *regression coefficients* computed using solely data points from the cluster $A^j, \; j = 1, \dots, k$.

3.1 SVM in linear regression

We start with the brief description of the SVM approach for linear regression, where the aim is to fit one hyperplane f (linear function) into the given data set A with a precision $\varepsilon > 0$. Therefore, in ε-SVM linear regression we define the function f as follows [8, 31]

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

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.

The above mentioned problem can be formulated as the following nonsmooth optimization problem

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

To obtain a solution for (2), we need to have at least one feasible point. This, however, 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, problem (2) can be reformulated as an unconstrained convex nonsmooth optimization problem

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

From this formulation it is easy to see that we can tolerate small deviations from hyperplanes since they are not penalized. This gives us more "freedom" in the solution process since all points are acceptable. In practice, it is common to have some noise in data sets and, due to this, it is more reasonable to allow small perturbations from the hyperplanes.

3.2 Formulation of SVM-CLR problem

In SVM-CLR, 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 $\{\boldsymbol{x}^1, y_1\}, \dots, \{\boldsymbol{x}^k, y_k\}$ where $\boldsymbol{x}^i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$ and, in what follows, the vectors $\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^k)^T \in \mathbb{R}^{nk}$ and $\boldsymbol{y} = (y_1, y_2, \dots, y_k)^T \in \mathbb{R}^k$ represent the combined values of these coefficients. Moreover, a point $(\boldsymbol{a}^i, b_i) \in A$ is associated with the closest hyperplane. In the spirit of (2), the *SVM-CLR problem* can be formulated as follows

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

By utilizing the same strategy as in the previous subsection, we are able to write the unconstrained nonsmooth version of the SVM-CLR problem as

$$\begin{cases} \min & F_k(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^{nk},\ \boldsymbol{y} \in \mathbb{R}^k \end{cases} \tag{4}$$

with the objective function

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \max\left(0, \min_{j=1,\dots,k} \left|(\boldsymbol{x}^j)^T\boldsymbol{a}^i + y_j - b_i\right| - \varepsilon\right). \tag{5}$$

It is worth noting that unlike (3) problem (4) is nonconvex since we fit more than one function at a time.

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 by* (5) *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

$$F_k^1(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \max\left\{\sum_{j=1}^{k} e_i(\boldsymbol{x}^j, y_j),\ \max_{j=1,\dots,k} \sum_{t=1, t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon\right\}$$

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

Proof. Consider the function

$$\psi_i(\boldsymbol{x}, \boldsymbol{y}) = \min_{j=1,\dots,k} e_i(\boldsymbol{x}^j, y_j) \quad \text{for } i = 1, \dots, m,$$

which is a minimum of convex piecewise linear functions. The function ψ_i is of the form presented in the case (i) of Lemma 2.3 since the convex function e_i can be presented as a DC function $f = f^1 - f^2$, where $f^1 = e_i$ and $f^2 = 0$. 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

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

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

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

which can be represented as a difference of two 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

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

Therefore, the function F_k can be rewritten as

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

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

$$F_k^1(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \varphi_i^1(\boldsymbol{x},\boldsymbol{y}) \quad \text{and} \quad F_k^2(\boldsymbol{x},\boldsymbol{y}) = C\sum_{i=1}^{m} \varphi_i^2(\boldsymbol{x},\boldsymbol{y}).$$

This completes the proof. □

It is worth to note that each DC function has an infinite number of different DC decompositions. The DC decomposition of the objective F_k presented in Proposition 3.1 is only one option among the set of possible ones. Since the selected DC decomposition can affect the performance of the algorithm we present an alternative DC representation for F_k.

Proposition 3.2. *For the DC function F_k defined by* (5)*, an alternative DC decomposition is*

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

with the DC components

$$\tilde{F}_k^1(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m}\sum_{j=1}^{k} \max\Big\{0,\, e_i(\boldsymbol{x}^j, y_j) - \varepsilon\Big\} \ \textit{and}$$

$$\tilde{F}_i^2(\boldsymbol{x},\boldsymbol{y}) = C\sum_{i=1}^{m} \max_{j=1,\dots,k} \sum_{t=1,t\neq j}^{k} \max\Big\{0,\, e_i(\boldsymbol{x}^t, y_t) - \varepsilon\Big\}.$$

Proof. The function F_k, defined in (5), can be rewritten as

$$\tilde{F}_k(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \min_{j=1,\dots,k} \Big\{ \max\Big\{0,\, e_i(\boldsymbol{x}^j, y_j) - \varepsilon\Big\}\Big\},$$

which is an alternative representation for the objective function F_k. To prove this we show that

$$\min_{j=1,\dots,k} \Big\{ \max\Big\{0, e_i(\boldsymbol{x}^j, y_j) - \varepsilon\Big\}\Big\} = \max\Big\{0, \min_{j=1,\dots,k} e_i(\boldsymbol{x}^j, y_j) - \varepsilon\Big\}. \qquad (6)$$

It is clear that both sides of (6) are non-negative. Assume first $\min_{j=1,\dots,k} e_i(\boldsymbol{x}^j, y_j) \leq \varepsilon$ which implies that there exists at least one index $l \in \{1,\dots,k\}$ such that $\max\{0, e_i(\boldsymbol{x}^l, y_l) - \varepsilon\} = 0$. This yields

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

On the other hand, if $\min_{j=1,\dots,k} e_i(\boldsymbol{x}^j, y_j) > \varepsilon$, then we obtain

$$\begin{aligned}\min_{j=1,\dots,k} \Big\{ \max\{0, e_i(\boldsymbol{x}^j, y_j) - \varepsilon\} \Big\} &= \min_{j=1,\dots,k} \Big\{ e_i(\boldsymbol{x}^j, y_j) - \varepsilon \Big\} \\ &= \max \Big\{ 0, \min_{j=1,\dots,k} e_i(\boldsymbol{x}^j, y_j) - \varepsilon \Big\}.\end{aligned}$$

Therefore, equality (6) is true and the functions $\tilde{F}_k$ and F_k are identical.

In addition, using Lemma 2.3 we notice that for the function

$$g_i(\boldsymbol{x}, \boldsymbol{y}) = \min_{j=1,\dots,k} \Big\{ \max \Big\{ 0,\ e_i(\boldsymbol{x}^j, y_j) - \varepsilon \Big\} \Big\}$$

the DC decomposition is

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

where the DC components are selected as

$$\begin{aligned} g_i^1(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{j=1}^{k} \max \Big\{ 0,\ e_i(\boldsymbol{x}^j, y_j) - \varepsilon \Big\} \quad \text{and} \\ g_i^2(\boldsymbol{x}, \boldsymbol{y}) &= \max_{j=1,\dots,k} \sum_{t=1, t\neq j}^{k} \max \Big\{ 0,\ e_i(\boldsymbol{x}^t, y_t) - \varepsilon \Big\}. \end{aligned}$$

Using this result in the alternative representation $\tilde{F}_k$ of the objective F_k, we can write the DC decomposition of F_k in the form

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

where

$$\tilde{F}_k^1(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C \sum_{i=1}^{m} g_i^1(\boldsymbol{x}, \boldsymbol{y}) \quad \text{and} \quad \tilde{F}_i^2(\boldsymbol{x}, \boldsymbol{y}) = C \sum_{i=1}^{m} g_i^2(\boldsymbol{x}, \boldsymbol{y}).$$

$\square$

3.3 Auxiliary SVM-CLR problem

Problem (4) is nonconvex and it may have many local solutions. The success of local search methods in finding global or near global 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 [3], but instead of CLR problems, it is designed for SVM-CLR problems. The basic idea in the incremental approach is that the solution for the SVM-CLR problem with $k-1$ hyperplanes can be used to derive a good starting point for the SVM-CLR problem with k hyperplanes. Since the solution is build incrementally we next present the so-called auxiliary problem having 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 of the SVM-CLR problem with $k-1$ hyperplanes. The *regression error* of the data point $(\boldsymbol{a}^i, b_i) \in A$ is denoted by

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

The *kth auxiliary SVM-CLR problem* is

$$\begin{cases} \min & \bar{F}_k(\boldsymbol{u}, v) \\ \text{s.t.} & \boldsymbol{u} \in \mathbb{R}^n, \ v \in \mathbb{R} \end{cases} \tag{7}$$

with the objective funtion

$$\bar{F}_k(\boldsymbol{u}, v) = \frac{1}{2}\|\boldsymbol{u}\|^2 + \sum_{i=1}^m \min\Big\{r^i_{k-1}, \max\Big\{0, e_i(\boldsymbol{u}, v) - \varepsilon\Big\}\Big\}. \tag{8}$$

It is worth noting that, if $r^i_{k-1} = 0$ for $(\boldsymbol{a}^i, b_i) \in A$ then this point $(\boldsymbol{a}^i, b_i)$ can be omitted from problem (7). Thus, the only interesting points in the auxiliary problem are those for which $r^i_{k-1} > 0$. In addition, the auxiliary SVM-CLR problem (7) is much easier and less time-consuming to solve than the original SVM-CLR problem (4) due to the less number of variables.

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

Proposition 3.3. *Let $\bar{F}_k$ be the function defined in* (8)*. Then $\bar{F}_k$ is a DC function and its DC decomposition can be written in the form*

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

where the DC components are

$$\bar{F}^1_k(\boldsymbol{u}, v) = \frac{1}{2}\|\boldsymbol{u}\|^2 + \sum_{i=1}^m \Big(r^i_{k-1} + \max\Big\{0, e_i(\boldsymbol{u}, v) - \varepsilon, \Big\}\Big) \quad \textit{and}$$

$$\bar{F}^2_k(\boldsymbol{u}, v) = \sum_{i=1}^m \max\Big\{r^i_{k-1}, \max\Big\{0, e_i(\boldsymbol{u}, v) - \varepsilon\Big\}\Big\}.$$

Proof. The DC decomposition is obtained by noticing that in (8) the term $\min\left\{r^i_{k-1}, \max\{0, e_i(\boldsymbol{u}, v) - \varepsilon\}\right\}$ is a minimum of convex functions. Thus, we can apply the case (i) of Lemma 2.3. This yields directly the result. □

4 Double bundle method for SVM-CLR problems

In this section, we present a modified double bundle method (DB-SVM-CLR) to solve SVM-CLR problems. The main idea in the new DB-SVM-CLR method is to combine the best features of the double bundle method (DBDC), [18] and the incremental algorithm [3] using the SVM-CLR formulation. The DBDC is a local solution method for nonsmooth DC optimization, which utilizes explicitly a DC decomposition of the objective function to take advantage of both the convexity and 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 SVM-CLR problem (4) and the auxiliary SVM-CLR problem (7) at each iteration of the incremental algorithm. A more detailed description of the algorithms follow.

4.1 DBDC method

For simplicity, we describe the DBDC method for a DC function f defined on the n-dimensional space $\mathbb{R}^n$. The DC structure 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.

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

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

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

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

for the DC component f^i, $i = 1, 2$. This model is the classical one used in convex bundle methods (see e.g., [20, 24, 30]) and it supports from below the epigraph of a convex function.

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 concave behaviour of f.

The approximation of f is used to determine the search direction $\boldsymbol{d}_t \in \mathbb{R}^n$. Therefore, at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ we need to globally solve the following nonsmooth nonconvex DC minimization problem

$$\begin{cases} \min & \hat{f}(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s. t.} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{9}$$

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 problem (9), 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 [21, 22] and utilized in [17].

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

$$f(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k) \leq m\left(\hat{f}(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k)\right) \tag{10}$$

needs to be satisfied, where $\hat{f}(\boldsymbol{x}_k + \boldsymbol{d}_t) - f(\boldsymbol{x}_k) < 0$ is the predicted descent and $m \in (0, 1)$ is the descent parameter. This guarantees that the value of the objective f decreases sufficiently and, thus, we can update the iteration point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_t$. Otherwise, if condition (10) does not hold, the used model is not accurate enough to yield a descent 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 criteria 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 escaping procedure [18] 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 whole algorithm terminates. Otherwise, a solution candidate $\boldsymbol{x}_k$ is not Clarke stationary. In this case, the escaping procedure generates a descent direction and we apply the DBDC method starting from a new better point $\boldsymbol{x}^+$.

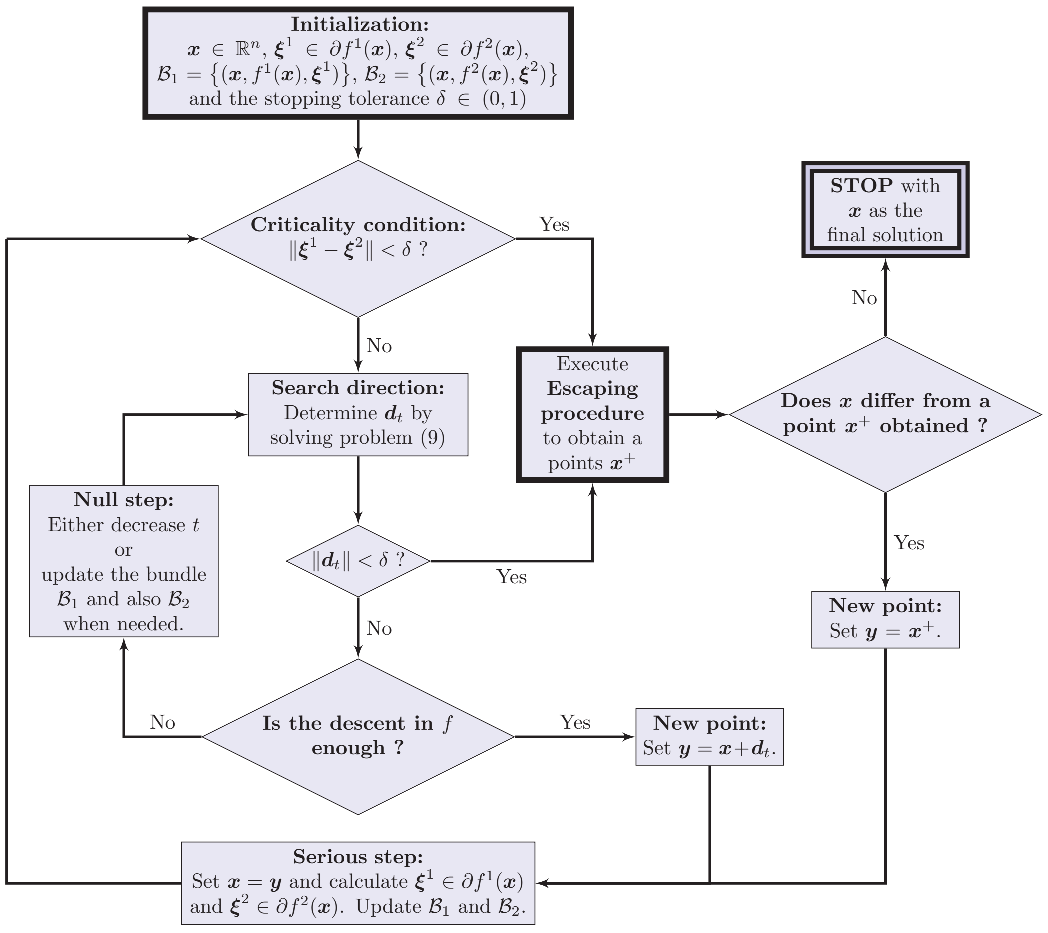

Initialization:
$\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\xi}^1 \in \partial f^1(\boldsymbol{x})$, $\boldsymbol{\xi}^2 \in \partial f^2(\boldsymbol{x})$,
$\mathcal{B}_1 = \{(\boldsymbol{x}, f^1(\boldsymbol{x}), \boldsymbol{\xi}^1)\}$, $\mathcal{B}_2 = \{(\boldsymbol{x}, f^2(\boldsymbol{x}), \boldsymbol{\xi}^2)\}$
and the stopping tolerance $\delta \in (0,1)$
Criticality condition:
$\|\boldsymbol{\xi}^1 - \boldsymbol{\xi}^2\| < \delta$?
Yes
No
STOP with $\boldsymbol{x}$ as the final solution
No
Search direction:
Determine $\boldsymbol{d}_t$ by solving problem (9)
Execute Escaping procedure to obtain a points $\boldsymbol{x}^+$
Does $\boldsymbol{x}$ differ from a point $\boldsymbol{x}^+$ obtained ?
Null step:
Either decrease t
or
update the bundle $\mathcal{B}_1$ and also $\mathcal{B}_2$ when needed.
$\|\boldsymbol{d}_t\| < \delta$?
Yes
Yes
No
New point:
Set $\boldsymbol{y} = \boldsymbol{x}^+$.
No
Is the descent in f enough ?
Yes
New point:
Set $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{d}_t$.
Serious step:
Set $\boldsymbol{x} = \boldsymbol{y}$ and calculate $\boldsymbol{\xi}^1 \in \partial f^1(\boldsymbol{x})$ and $\boldsymbol{\xi}^2 \in \partial f^2(\boldsymbol{x})$. Update $\mathcal{B}_1$ and $\mathcal{B}_2$.

Figure 1: DBDC method

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 SVM-CLR and auxiliary SVM-CLR problems.

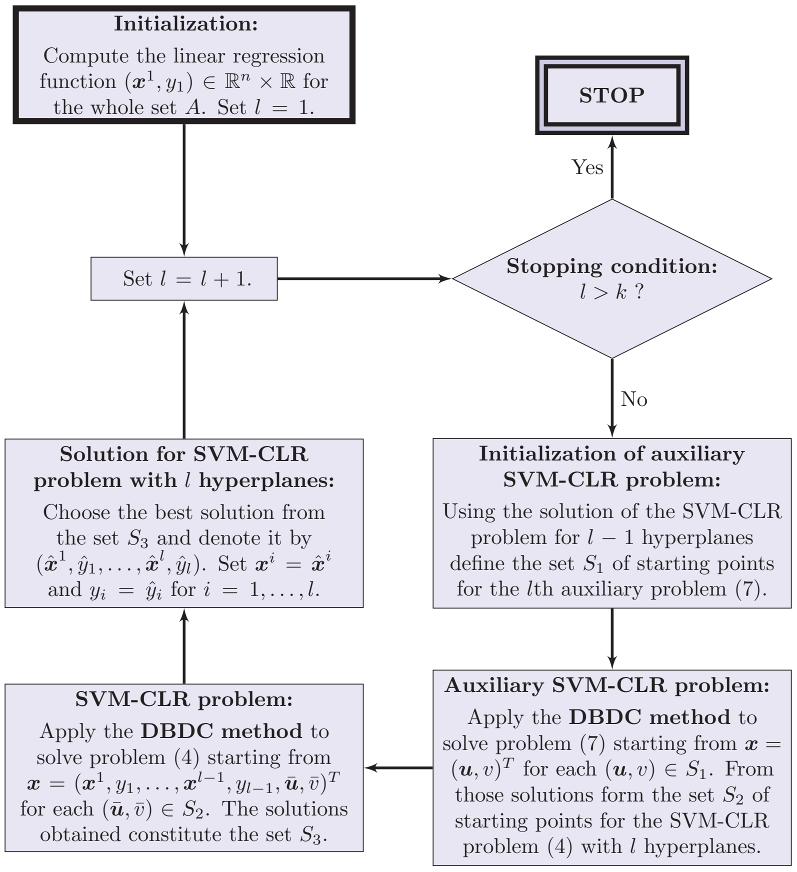
Lemma 4.1. *[18] Let the assumptions **A1** and **A2** be valid. During the DBDC, the execution of the escaping procedure stops after a finite number of iterations.*

Theorem 4.2. *[18] 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*

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

4.2 Incremental algorithm

Next, we introduce the incremental algorithm based on the method presented in [3], but instead of CLR problems, it is modified to SVM-CLR problems. The incremental algorithm starts with the calculation of one hyperplane approximating the whole data and gradually adds one hyperplane at each iteration until the required number of hyperplanes is calculated. During each iteration this method also utilizes the auxiliary problem (7) to generate a set of promising starting points for the SVM-CLR problem (4), since the auxiliary problem is a lot easier and less time-consuming to solve than (4).



Initialization:
Compute the linear regression function $(\boldsymbol{x}^1, y_1) \in \mathbb{R}^n \times \mathbb{R}$ for the whole set A. Set $l = 1$.
STOP
Yes
Set $l = l + 1$.
Stopping condition: $l > k$?
No
Solution for SVM-CLR problem with l hyperplanes:
Choose the best solution from the set S_3 and denote it by $(\hat{\boldsymbol{x}}^1, \hat{y}_1, \dots, \hat{\boldsymbol{x}}^l, \hat{y}_l)$. Set $\boldsymbol{x}^i = \hat{\boldsymbol{x}}^i$ and $y_i = \hat{y}_i$ for $i = 1, \dots, l$.
Initialization of auxiliary SVM-CLR problem:
Using the solution of the SVM-CLR problem for $l-1$ hyperplanes define the set S_1 of starting points for the lth auxiliary problem (7).
SVM-CLR problem:
Apply the DBDC method to solve problem (4) starting from $\boldsymbol{x} = (\boldsymbol{x}^1, y_1, \dots, \boldsymbol{x}^{l-1}, y_{l-1}, \bar{\boldsymbol{u}}, \bar{v})^T$ for each $(\bar{\boldsymbol{u}}, \bar{v}) \in S_2$. The solutions obtained constitute the set S_3.
Auxiliary SVM-CLR problem:
Apply the DBDC method to solve problem (7) starting from $\boldsymbol{x} = (\boldsymbol{u}, v)^T$ for each $(\boldsymbol{u}, v) \in S_1$. From those solutions form the set S_2 of starting points for the SVM-CLR problem (4) with l hyperplanes.

Figure 2: Incremental algorithm in the DB-SVM-CLR method

An algorithm based on the incremental algorithm and the DBDC method is called DB-SVM-CLR. Therefore, this algorithm constructs clusters as well as linear functions approximating them incrementally. This means that the DB-SVM-CLR method do not solve just the SVM-CLR problem with k hyperplanes, but yields as the by-product solutions for each SVM-CLR problem with a smaller number of hyperplanes.

The detailed description of the method is presented in Figure 2, where S_1 denotes the set of starting points used at the initialization step for solving the auxiliary SVM-CLR problem. This set is selected to be quite large whereas the set S_2 used to solve the SVM-CLR 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 SVM-CLR problem (4). The detailed description of the selection of the sets S_1, S_2 and S_3 is presented in [3].

5 Numerical results

The aim of this section is to present some numerical results to prove the usability of the SVM-CLR formulation (5) to solve the CLR problem. Therefore, we illustrate the suitability of the SVM-CLR model and compare results with a frequently used fit function. This way we are able to show the main differences between these two functions and motivate the usage of the new SVM-CLR formulation.

In the comparisons, we use the piecewise quadratic fit function

$$\hat{F}_k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \min_{j=1,\dots,k} \left\{ ((\boldsymbol{x}^j)^T \boldsymbol{a}^i + y_j - b_i)^2 \right\}. \tag{11}$$

This fit function is frequently applied to solve CLR problems, for example, in [3, 4, 19]. In order to minimize the fit function (11), we apply the LMBM-CLR method [19] being designed to solve large-scale CLR problems and it combines the incremental algorithm [3] and the limited memory bundle method [14, 15]. Since the value of the fit function (11) is not comparable with that of the SVM-CLR model, we illustrate results by drawing the solution hyperplanes in both cases and then compare them visually.

Algorithms are tested using six different data sets. Three of them are generated using known hyperplanes and the others are generated clusterwise. All data sets have one numeric input and output variable to allow visualization of results. The number of data points ranges from 100 to 1420. The brief description of data sets is given in Table 1, where m stands for the number of data points and n for the number of input variables. We use k for the number of linear regression functions (or clusters).

The codes `DB-SVM-CLR` and `LMBM-CLR` of the above methods are implemented in Fortran 95 and the subroutine `PLQDF1` [23] is used in `DB-SVM-CLR` to

Table 1: The brief description of the data sets

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

solve the quadratic problem (9). Both codes are compiled by using `gfortran`, the GNU Fortran compiler, and performed on an Intel® Core™ i5-2400 CPU (3.10GHz, 3.10GHz) running on Windows 7.

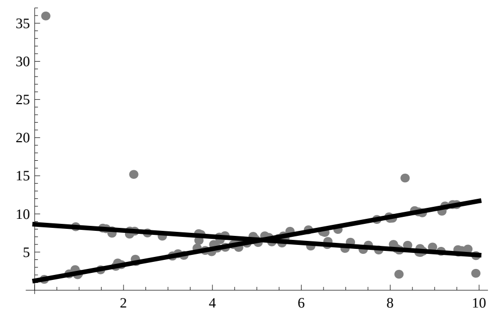
In the following, the numerical results of `DB-SVM-CLR` are obtained by using the DC decomposition presented in Proposition 3.1. Since the DC decomposition can affect the performance of the method we calculated the results also with the other DC decomposition stated in Proposition 3.2. Those results did not visually differ much from the ones presented here and, thus, they are not included.

The parameters of `DB-SVM-CLR` are selected to be as follows: the regularization parameter $C = 1$ and the stopping tolerance

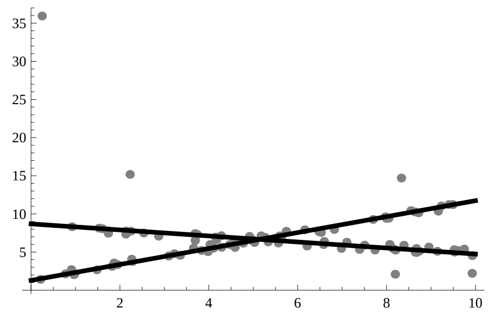
$$\delta = \begin{cases} 10^{-4}, & \text{for problem (7)} \\ 10^{-3}, & \text{for problem (4).} \end{cases}$$

The size of $\mathcal{B}_1$ is set to 50. For $\mathcal{B}_2$ the size is one for problem (7) and 3 otherwise. In the escaping procedure, we also use a bundle and its size is set to 100. For the other parameters, we apply the default values [18], but we set the value 10^{11} for the increase parameter R (see [18] for details). In `LMBM-CLR`, we use the default values given in [19].

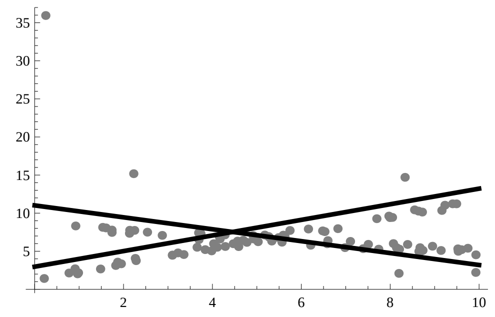
In Figure 3, the performance of the SVM-CLR formulation (5) on the parameter ε is illustrated, and Two Lines data set is used for this purpose. This data set is created by using two known lines and adding some random noise and a couple of distinct outliers. When ε is close to zero, for example 0.5, we cannot yet observe a difference from the solution with $\varepsilon = 0.0$. Moreover, in both cases $\varepsilon = 0.0$ and $\varepsilon = 0.5$, we find hyperplanes describing the data correctly. However, in the cases $\varepsilon = 2.0$ and $\varepsilon = 3.0$, we see that the outliers have a significant influence. Thus, the parameter ε cannot be too large since this can also completely distort the solution as is seen in Figure 3 when $\varepsilon = 3.0$. Since $\varepsilon = 0.5$ is not affected by outliers this justifies the usage of a small positive $\varepsilon > 0$ allowing us to have more freedom in the solution process. Therefore, in the rest of this section we use the value $\varepsilon = 0.5$.



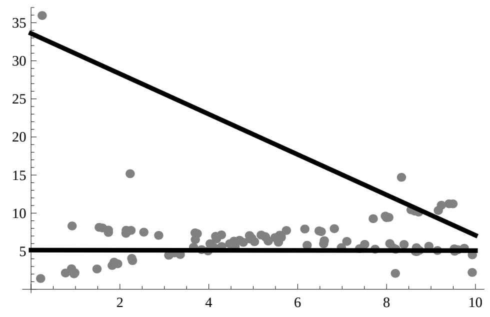
a) The solution for $\varepsilon = 0.0$



b) The solution for $\varepsilon = 0.5$



c) The solution for $\varepsilon = 2.0$



d) The solution for $\varepsilon = 3.0$

Figure 3: Results for Two Lines data set with the SVM-CLR model (5), $k = 2$ and different values of the parameter ε

The solution for Two Lines data set with $k = 2$ using the fit function (11) 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 fit function (11) does not lead to the finding of correct hyperplanes since two outliers distort the solution. Thus, in this example the SVM-CLR formulation (5) is more reliable than (11).

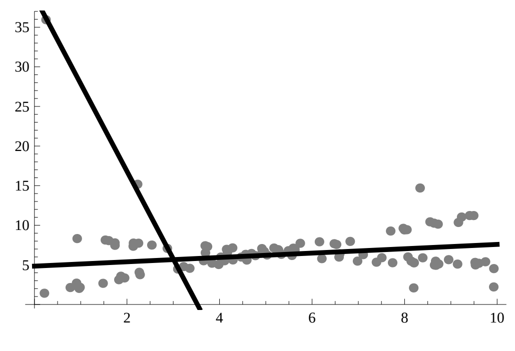
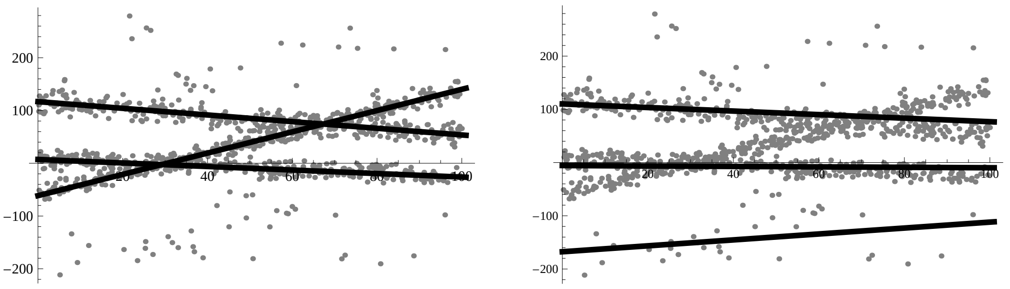


Figure 4: Result for Two Lines data set with the fit function (11) for $k = 2$



a) The SVM-CLR model (5) b) The fit function (11)

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

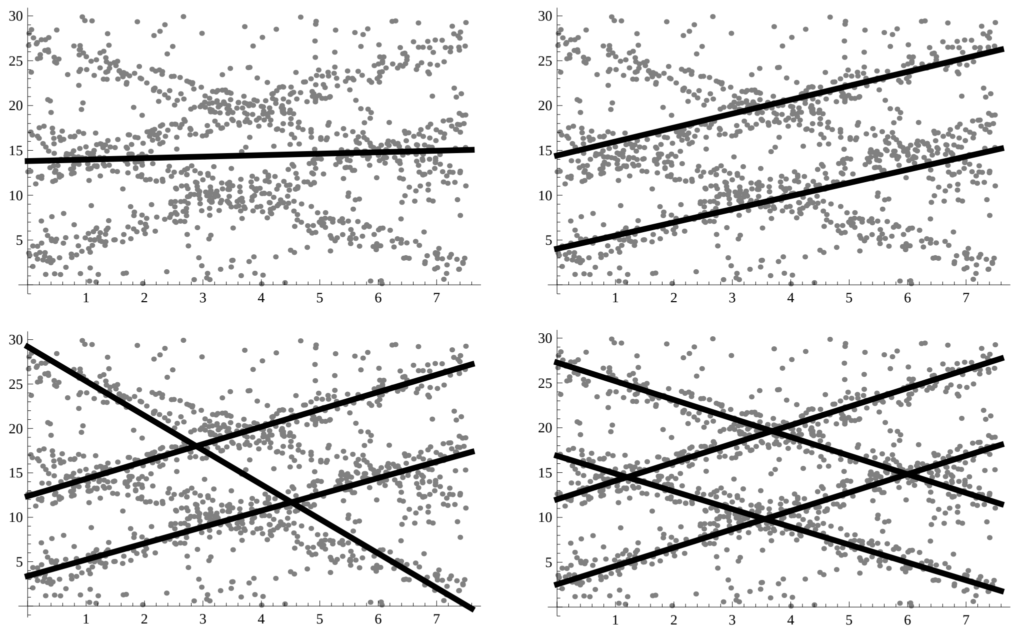


Figure 6: Results for Four Lines data set with the SVM-CLR model (5) and different k

The results for Three Lines data set with both the new model (5) and the fit function (11) are presented in Figure 5. This data set is generated based on three different lines by adding some noise and outliers. The results show that the fit function finds two out of three lines and the third line approximates the outliers. However, the new SVM-CLR formulation manages to distinguish correctly all the hyperplanes from the outliers. Thus, this example also shows that the new model has ability to distinguish outliers.

In Figure 6, we present the solutions to Four Lines data set with the new model (5) and different number of hyperplanes $k = 1, 2, 3$ and 4. Similar to the previous example, this data set is generated by using four known

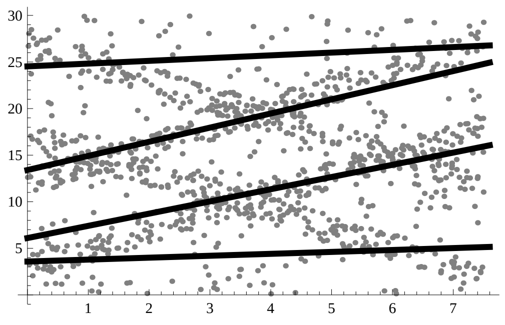


Figure 7: Result for Four Lines data set with the fit function (11) and $k = 4$

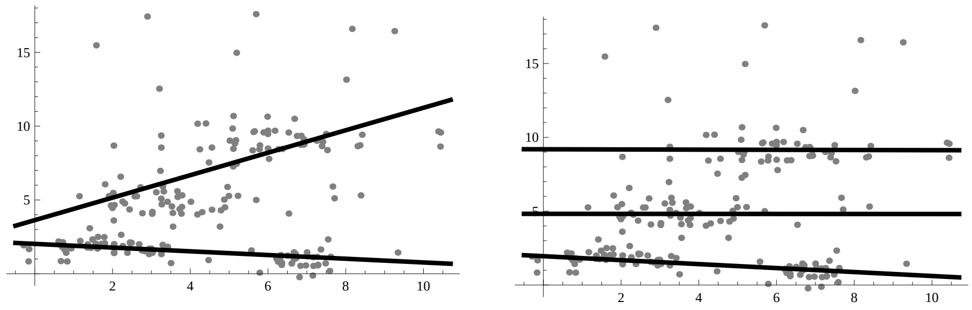


Figure 8: Result for Clusters 1 data set with the SVM-CLR model (5) and $k = 2, 3$

hyperplanes and adding some random noise and outliers. This example shows that the final number of hyperplanes should be chosen carefully since it affects the final result and how well it describes the data. However, in this example even in the cases $k = 2$ and $k = 3$, the use of the SVM-CLR formulation allows to correctly identify two hyperplanes.

The solution for Four Lines data set with the fit function (11) and $k = 4$ is presented in Figure 7. This example shows that the fit function can have great difficulties when data points cover the input space quite evenly. Now, 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 the data evenly.

Next, we consider Clusters 1 data set, where there are four clusters with some outliers. The solutions for the SVM-CLR formulation (5) and the fit function (11) are presented in Figures 8 and 9, respectively. In the case $k = 2$, the fit function provides an inaccurate solution since one hyperplane is positioned between clusters and fails to give any useful information. However, the SVM-CLR formulation provides quite an usable solution in the case $k = 2$ since each of the hyperplanes captures an approximation for two clusters.

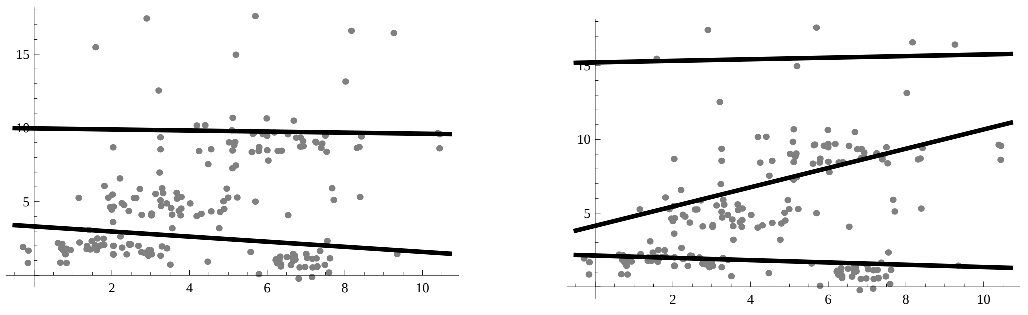
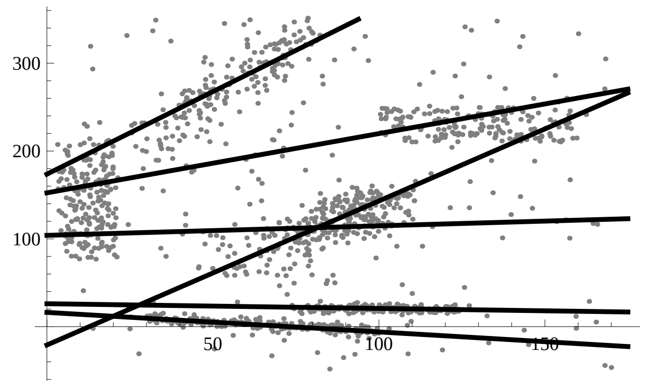
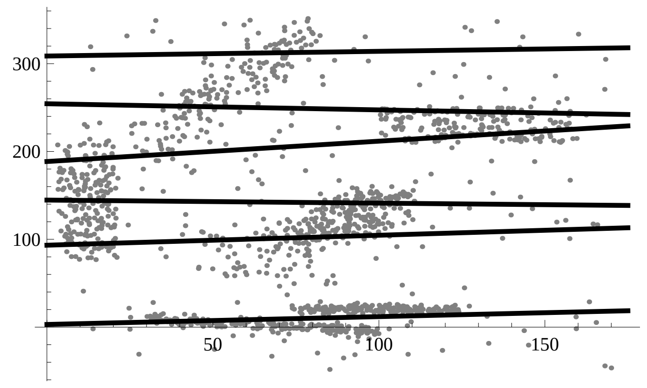


Figure 9: Results for Clusters 1 data set with the fit function (11) and $k = 2, 3$

Moreover, in the case $k = 3$ the solution with the SVM-CLR formulation describes the data quite correctly, whereas the fit function (11) produces a solution where one hyperplane only approximates the outliers.



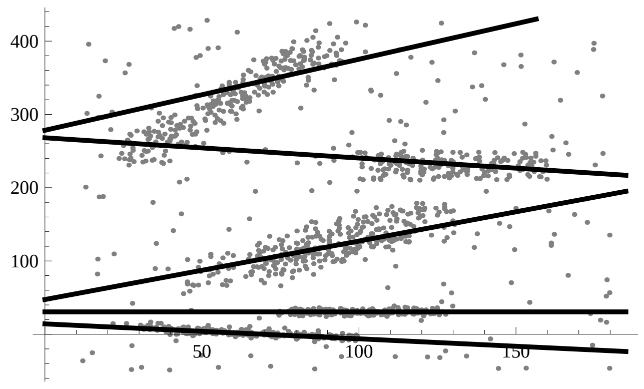
a) The SVM-CLR model (5)



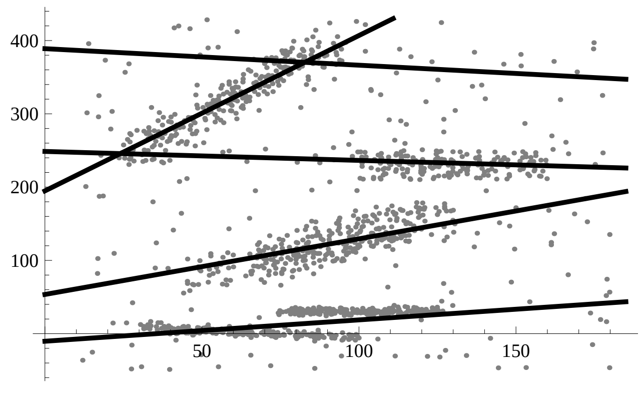
b) The fit function (11)

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

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 approximate since the slope of the hyperplane for this cluster is close to infinity. Moreover, in the SVM-CLR formulation (5) we minimize the norm of the slope of the hyperplane and, therefore, vertical hyperplanes are not possible to detect. In spite of that, the SVM-CLR model is able to find correctly four out of six hyperplanes. The vertical cluster is covered with three different hyperplanes and this also affects the fact that we do not find the best approximation for one horizontal cluster. Results for this data set show that the fit function (11) can be sensitive to vertical clusters since the solution just consists of horizontal hyperplanes covering the area of the data evenly. Thus, the horizontal clusters are represented in the solution, but the structure of the other clusters is not captured.



a) The SVM-CLR model (5)



b) The fit function (11)

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

In Figure 11, we illustrate the solutions for Clusters 3 data set. This data set resembles Clusters 2 data set, but it does not contain a vertical cluster. Based on the pictures it is not possible to say which one of the solutions is better. For example, the solution obtained with the SVM-CLR formulation (5) described quite accurately the given data. However, for one cluster (left top corner) the hyperplane is not accurate since the approximation of the other clusters and the outliers distort the solution slightly. For fit function (11), the solution is also quite accurate, but one hyperplane 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 the fit function and SVM-CLR formulation are not able to detect this kind of clusters.

6 Conclusions

In this paper, we have introduced a new SVM-CLR formulation to solve the clusterwise linear regression (CLR) problem. The novelty in the formulation is that the support vector machines (SVM) approach is incorporated into the clusterwise linear regression problem. This way we have been able to obtain more flexible and reliable model. The new SVM-CLR formulation is presented as a difference of convex (DC) functions in order to capture both the convexity and the concavity of the objective function.

In addition, we have introduced a new DB-SVM-CLR method to solve the CLR problem designed by using the new SVM-CLR formulation. The DB-SVM-CLR combines the double bundle method DBDC for nonsmooth DC optimization with the incremental algorithm utilizing the SVM-CLR formulation. This combination enables us to find global or near global solutions since solutions are constructed incrementally by solving the SVM-CLR problems and the auxiliary SVM-CLR problems with the DBDC method.

To validate the usability of the SVM-CLR formulation, we have tested the proposed DB-SVM-CLR method by using six different data sets. Three data sets are generated by using known linear functions and three others using known clusters. Such a choice of data sets allow us to demonstrate the ability to detect linear functions for data sets having different data structures. Numerical results show that the DB-SVM-CLR method is efficient and reliable to solve the CLR problems since it nearly always produces solutions describing the data accurately. The comparison with the frequently used piecewise quadratic fit function also supports this conclusion since the SVM-CLR model outperforms this fit function in most data sets used in numerical experiments. In addition, the new SVM-CLR formulation can tolerate outliers. This is a significant feature since outliers are common in data sets and they can easily distort the solution.

To conclude the numerical results show that the new DB-SVM-CLR algorithm developed for the new formulation of the CLR problem has a good performance. Nevertheless, there are still some open questions. First, it is not clear how efficient the new algorithm is as a prediction tool. Moreover, we have not considered the applicability of the DB-SVM-CLR for solving large-scale CLR problems. Finally, the generalization ability of the new algorithm is another important problem. All this will be the subject of the future research.

Acknowledgement. This work was funded by the University of Turku Graduate School UTUGS Matti Programme, the University of Turku, the Academy of Finland (Project No. 289500 and 294002) and the Australian Research Council's Discovery Projects funding scheme (Project No. DP140103213).

References

[1] A. M. Bagirov, N. Karmitsa, and M. M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software.* Springer, Cham, Heidelberg, 2014.

[2] A. M. Bagirov, A. Mahmood, and A. Barton. Prediction of monthly rainfall in Victoria, Australia: Clusterwise linear regression approach. *Atmospheric Research*, 188(Supplement C):20 – 29, 2017.

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

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

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

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

[7] F. H. Clarke. *Optimization and Nonsmooth Analysis.* Wiley-Interscience, New York, 1983.

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

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

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

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

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

[13] A. García-Escudero, A. Gordaliza, A. Mayo-Iscar, and R. San Martin. Robust clusterwise linear regression through trimming. *Computational Statistics and Data Analysis*, 54(12):3057–3069, 2010.

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

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

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

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

[18] K. Joki, A. M. Bagirov, N. Karmitsa, M. M. Mäkelä, and S. Taheri. Double bundle method for nonsmooth DC optimization. *TUCS Technical Report No. 1173, Turku Centre for Computer Science, Turku*, 2017.

[19] N. Karmitsa, A. M. Bagirov, and S. Taheri. Limited memory bundle method for solving large clusterwise linear regression problems. *TUCS*

Technical Report No. 1172, Turku Centre for Computer Science, Turku, 2016.

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

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

[22] H. A. Le Thi and T. Pham Dinh. 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):23–46, 2005.

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

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

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

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

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

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

[29] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[30] H. Schramm and J. Zowe. 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, 1992.

[31] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

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

[33] A. S. Strekalovsky. On local search in DC optimization problems. *Applied Mathematics and Computation*, 255:73 – 83, 2015.

[34] J. F. Toland. On subdifferential calculus and duality in nonconvex optimization. *Mémoires de la Société Mathématique de France*, 60:177–183, 1979.

[35] H. Tuy. *Convex Analysis and Global Optimization.* Kluwer Academic Publishers, Dordrecht, 1st edition, 1998.

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

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-3653-2
ISSN 1239-1891