TUCS

Napsu Karmitsa | Adil Bagirov | Sona Taheri

Limited Memory Bundle Method for Solving Large Clusterwise Linear Regression Problems

Turku Centre for Computer Science

TUCS Technical Report
No 1172, December 2016

Limited Memory Bundle Method for Solving Large Clusterwise Linear Regression Problems

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

Adil 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

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 1172, December 2016

Abstract

A clusterwise linear regression problem consists of finding a number of linear functions each approximating a subset of the given data. In this paper, the limited memory bundle method [Haarala et.al. *Math. Prog.*, Vol. 109, No. 1, pp. 181–205, 2007] is modified and combined with the incremental approach to solve this problem using its nonsmooth optimization formulation. The proposed algorithm is tested on small and large real world data sets and compared with other algorithms for clusterwise linear regression. Numerical results demonstrate that the proposed algorithm is especially efficient in data sets with large numbers of instances and input variables.

Keywords: Clusterwise linear regression; Nonsmooth optimization; Nonconvex problems; Bundle methods; Limited memory methods.

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

The clusterwise regression is a technique to approximate the data using two or more regression functions. It is based on two well-known techniques: clustering and regression, and simultaneously identifies clusters and their associated regression functions. If the regression functions are linear then the clusterwise regression is called *the clusterwise linear regression (CLR)*. The CLR has many applications (see, e.g. [26, 32]).

In general, the algorithms for the CLR can be divided into three groups. The first group contains algorithms which are extensions of clustering algorithms such as k-means [27, 28] and EM [14]. Algorithms from the second group are based on mixture models [11, 15, 26]. The third group consists of algorithms which are extensions of optimization methods. This group includes the simulated annealing method for the CLR [12], algorithms based on mixed integer nonlinear programming [8, 9, 11] and nonsmooth optimization (NSO) methods [1, 2, 3].

The CLR is a global optimization problem. However, it is out of the reach of existing global optimization algorithms when the number of input variables is relatively large and/or a large number of linear regression functions are needed to approximate data. Therefore, it is essential to develop algorithms which are capable of finding high quality solutions to the CLR problems in data sets with large numbers of points and/or input variables.

In this paper, we introduce the new LMBM-CLR -method for solving CLR problems. The LMBM-CLR -method consist of two algorithms: an incremental algorithm is used to solve CLR problems globally and at each iteration of this algorithm the limited memory bundle algorithm (LMBM) by Karmitsa (née Haarala) et al. [16, 17, 18, 20] is used to solve both the CLR problem and the so-called auxiliary CLR problem with different starting points provided by the incremental algorithm.

The LMBM is a modification of the variable metric bundle methods (VMBM) [24, 31], where the limited memory approach (see e.g. [6]) is used to calculate the search direction. Therefore, the time-consuming quadratic direction finding problem appearing in the standard bundle methods (see, e.g. [19, 22, 25]) does not need to be solved, nor the number of stored subgradients needs to grow with the dimension of the problem. Furthermore, the method uses only a few vectors to represent the variable metric approximation of the Hessian matrix and, thus, it avoids storing and manipulating large matrices as is the case in the VMBM. These improvements make the LMBM suitable for large-scale optimization. Namely, the number of operations needed for the calculation of the search direction is only linearly dependent on the number of variables while, for example, this dependence is quadratic for the VMBM . In this paper, the original LMBM algorithm is slightly modified to be better suited for solving CLR problems.

The rest of this paper is organized as follows. The nonsmooth optimization formulation of the CLR problem is given in Section 2. In Section 3, we first give the basic ideas of the LMBM and then, we recall the ideas of incremental approach used to solve globally the CLR problem. The results of the numerical experiments are presented and discussed in Section 4, and finally, Section 5 concludes the paper.

Throughout the paper the following notations are used: the Euclidean norm in $\mathbb{R}^n$ is denoted by $\|\cdot\|$ and the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted by $\boldsymbol{a}^T\boldsymbol{b}$ (bolded symbols are used for vectors).

2 Clusterwise Linear Regression

The aim of the CLR is to find an optimal partition of the given data set $A = \{(\boldsymbol{a}_i, b_i) \in \mathbb{R}^n \times \mathbb{R} : \ i = 1, \dots, m\}$ into k clusters and, simultaneously, to find regression coefficients $\{\boldsymbol{x}_j, y_j\}, \ \boldsymbol{x}_j \in \mathbb{R}^n, \ y_j \in \mathbb{R}, \ j = 1, \dots, k$ within clusters in order to minimize the overall fit. Let $A^j \subset A, \ j = 1, \dots, k$ be clusters such that

1. $A^j \neq \emptyset, j = 1, \dots, k;$
2. $A^j \bigcap A^l = \emptyset$, for all $j, l = 1, \dots, k, j \neq l;$
3. $A = \bigcup\limits_{j=1}^{k} A^j.$

Let $\{\boldsymbol{x}_j, y_j\}$ be linear regression coefficients computed using solely the data points from the cluster $A^j, \ j = 1, \dots, k$. Then for a given data point $(\boldsymbol{a}, b) \in A$ and coefficients $\{\boldsymbol{x}_j, y_j\}$ the squared regression error $E_{\boldsymbol{a}b}(\boldsymbol{x}_j, y_j)$ is given by

$$E_{\boldsymbol{a}b}(\boldsymbol{x}_j, y_j) = \big((\boldsymbol{x}_j)^T\boldsymbol{a} + y_j - b\big)^2.$$

A data point is associated with the cluster whose regression error at this point is the smallest one. The function

$$f_k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \min_{j=1,\dots,k} E_{\boldsymbol{a}b}(\boldsymbol{x}_j, y_j), \tag{1}$$

is called *the k-th clusterwise linear regression function or the k-th overall fit function* [1, 2, 3]. Here $\boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) \in \mathbb{R}^{nk}$ and $\boldsymbol{y} = (y_1, \dots, y_k) \in \mathbb{R}^k$. For $k = 1$ the function f_k is convex and for $k > 1$ it is nonsmooth, nonconvex, and piecewise quadratic.

The CLR problem. The *NSO formulation of the CLR problem* is given by

$$\begin{cases} \text{minimize} & f_k(\boldsymbol{x}, y) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) \in \mathbb{R}^{nk}, \ y \in \mathbb{R}^k, \end{cases} \tag{2}$$

where $f_k(\boldsymbol{x}, y)$ is defined in (1). The number of clusters k is not always known a priori and this number should be specified before solving Problem (2). The number of variables in Problem (2) is $(n + 1) \times k$ and it does not depend on m, the number of points in a data set.

The auxiliary CLR problem. Problem (2) is nonconvex and may have a large number of local solutions. In this paper, we propose to use the local method LMBM to solve it. The success of this algorithm strongly depends on the choice of initial solutions. We apply an algorithm introduced in [1] to generate such solutions. This algorithm uses the so-called auxiliary CLR problem. In this paper, we briefly recall this problem and refer to [1] for details.

Given the solution $(\boldsymbol{x}_1, y_1, \cdots, \boldsymbol{x}_{k-1}, y_{k-1}),\ k \geq 2$ to the $(k-1)$-CLR problem we define the regression error of the point $(\boldsymbol{a}, b) \in A$ by

$$r^{\boldsymbol{a}b}_{k-1} = \min_{j=1,\ldots,k-1} E_{\boldsymbol{a}b}(\boldsymbol{x}_j, y_j)$$

and introduce the following function

$$\bar{f}_k(\boldsymbol{u}, v) = \sum_{(\boldsymbol{a},b)\in A} \min\left\{r^{\boldsymbol{a}b}_{k-1}, E_{\boldsymbol{a}b}(\boldsymbol{u}, v)\right\},\ \boldsymbol{u} \in \mathbb{R}^n,\ v \in \mathbb{R}. \tag{3}$$

The function $\bar{f}_k$ is called *the k-th auxiliary clusterwise linear regression function* [1, 2, 3]. The problem

$$\begin{cases} \text{minimize} & \bar{f}_k(\boldsymbol{u}, v) \\ \text{subject to} & \boldsymbol{u} \in \mathbb{R}^n, v \in \mathbb{R}. \end{cases} \tag{4}$$

is called *the k-th auxiliary clusterwise linear regression problem* [1, 2, 3].

Similar to the k-CLR problem (2), the k-auxiliary CLR problem (4) is nonsmooth and nonconvex. However, the number of variables is only $n+1$ and it does not depend on the number of linear regression functions (clusters).

Initial Solutions. Since we apply the local method LMBM to solve problem (4) it is imperative to use an algorithm to generate good starting points to obtain high quality solutions. We apply an algorithm from [1] to find such points. This algorithm uses clusters from $(k-1)$-th iteration and computes hyperplanes passing through each data point and parallel to the hyperplane approximating the cluster to which this point belongs. Then all hyperplanes giving the value of the auxiliary CLR function less than some threshold are chosen as initial solutions to solve the problem (4). Details of this algorithm can be found in [1].

In their turn, solutions from the auxiliary CLR problem are used to compute initial solutions to solve the problem (2). The use of many initial solutions allows to get a set of solutions to the problem (4). The set of initial solutions to the problem (2) is obtained by simply adding each solution of the auxiliary problem to the solution $(\boldsymbol{x}_1, y_1, \cdots, \boldsymbol{x}_{k-1}, y_{k-1})$ to the $(k-1)$-CLR problem. Then the solutions providing the value of the CLR function less than some threshold are chosen as initial solutions to the problem (2). Such an approach allows to select most promising initial solutions and reduce computational effort.

3 LMBM-CLR -Method

In this section we introduce the new LMBM-CLR -method for solving CLR problems. As already said in the introduction, the LMBM-CLR -method consist of two algorithms: an incremental algorithm is used to solve CLR problems globally and at each iteration of this algorithm the LMBM is used to solve the CLR problem (2) and the auxiliary CLR problem (4).

3.1 LMBM

The LMBM is originally developed for solving general nonconvex nonsmooth optimization problems. Here, the original algorithm is slightly modified to be better suited for solving CLR problems. To use LMBM it is assumed that the objective function is locally Lipschitz continuous (LLC) and at every point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate both the value of the objective function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\{ \lim_{i\to\infty} \nabla f(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists} \},$$

where "conv" denotes the convex hull of a set. For (auxiliary) CLR problems these assumptions are easily satisfied. In this section our notation differs a little bit from that before: n is used as a size of the optimization problem, that is, $n = n + 1$ for auxiliary CLR problem and $n = (n + 1) \times l$ for CLR problem, where l is the current number of clusters. In addition, k is now used as an iteration counter.

The LMBM is characterized by the usage of null steps together with a simple aggregation of subgradients. Moreover, the limited memory approach is utilized in the calculation of the search direction and the aggregated values. The L-BFGS update formula is used after a serious step and the L-SR1 update formula after a null step. The usage of null steps gives further information about the nonsmooth objective in the case that the search direction is not "good enough". On the other hand, a simple aggregation of subgradients is used to guarantee the global convergence of the method.

Algorithm. Now we give the pseudo-code of LMBM for solving the (auxiliary) CLR problem. Here the following input is needed:

- $\boldsymbol{x}_1 \in \mathbb{R}^n$ — starting point;
- $\varepsilon > 0$ — stopping tolerance;
- $\hat{m}_c \geq 3$ — the maximum number of stored corrections used to form limited memory matrix updates.

```
PROGRAM LMBM
  INITIALIZE $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\hat{m}_c \geq 3$, and $\varepsilon > 0$;
  Compute $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$;
  Set $k = 1$, $m = 1$, $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$, $\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$, and $\tilde{\beta}_1 = 0$;
  WHILE the termination condition $w_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k + 2\tilde{\beta}_k \leq \varepsilon$ is not met
    Find step sizes $t_L^k$ and $t_R^k$, and the subgradient locality
      measure $\beta_{k+1}$;
    Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k$ and $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k$;
    Evaluate $f(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$;
    Store the new correction vectors $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ and
      $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$;
    Set $\hat{m}_k = \min\{k, \hat{m}_c\}$;
    IF $t_L^k > 0$ THEN
      SERIOUS STEP
        Compute the search direction $\boldsymbol{d}_{k+1}$ using $\boldsymbol{\xi}_{k+1}$ and L-BFGS
          update with $\hat{m}_k$ most recent correction pairs;
        Set $m = k + 1$ and $\tilde{\beta}_{k+1} = 0$;
      END SERIOUS STEP
    ELSE
      NULL STEP
        Determine multipliers $\lambda_i^k$ satisfying $\lambda_i^k \geq 0$ for all
        $i \in \{1,2,3\}$, and $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function
          $\varphi(\lambda_1,\lambda_2,\lambda_3) = [\lambda_1\boldsymbol{\xi}_m + \lambda_2\boldsymbol{\xi}_{k+1} + \lambda_3\tilde{\boldsymbol{\xi}}_k]^T D^k [\lambda_1\boldsymbol{\xi}_m + \lambda_2\boldsymbol{\xi}_{k+1} + \lambda_3\tilde{\boldsymbol{\xi}}_k]$
                    $+2(\lambda_2\beta_{k+1} + \lambda_3\tilde{\beta}_k)$;
        Compute the aggregate values
          $\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k\boldsymbol{\xi}_m + \lambda_2^k\boldsymbol{\xi}_{k+1} + \lambda_3^k\tilde{\boldsymbol{\xi}}_k$    and
          $\tilde{\beta}_{k+1} = \lambda_2^k\beta_{k+1} + \lambda_3^k\tilde{\beta}_k$;
        Compute the search direction $\boldsymbol{d}_{k+1}$ using $\tilde{\boldsymbol{\xi}}_{k+1}$ and L-SR1
          update with $\hat{m}_k$ most recent correction pairs;
      END NULL STEP
    END IF
    Set $k = k + 1$;
  END WHILE
  RETURN final solution $\boldsymbol{x}_k$;
END PROGRAM LMBM
```

REMARK 3.1. When combined with an incremental algorithm we use nonmonotone line search to find step sizes t_L^k and t_R^k. In addition, different stopping tolerances for different problems can be used.

Under the upper semi-smoothness assumption (see, e.g. [4]) the LMBM can be proved to be globally convergent for LLC objective functions [16, 18]. In addition, if we choose $\varepsilon > 0$, the LMBM algorithm terminates in a finite number of steps.

3.2 Incremental algorithm

Now we present the incremental algorithm for solving the CLR problem (2). Problem (2) is nonconvex and, it is important to use many starting points when applying a local method like LMBM for its solution. As already mentioned, we use the algorithm introduced in [1] to generate initial solutions in our algorithm given below. The LMBM is applied to solve both the CLR and the auxiliary CLR problems at each iteration of the incremental algorithm. Together, these two algorithms are a called the LMBM-CLR -method.

```
PROGRAM Incremental Algorithm
  INITIALIZE the maximum number of linear functions $k \geq 1$;
  Compute the linear regression function $(\boldsymbol{x}_1, y_1) \in \mathbb{R}^n \times \mathbb{R}$ of the
    set $A$;
  Set $l = 1$;
  WHILE $l < k$
    Set $l = l + 1$;
    Apply the procedure from [1] to find the set $S_1 \subset \mathbb{R}^{n+1}$ of
      initial solutions for the auxiliary CLR problem (4)
      with $k = l$;
    SOLVING AUXILIARY CLR PROBLEM
      To obtain a set $S_2 \subset \mathbb{R}^{n+1}$ of initial solutions for the
        $l$-CLR problem (2), apply LMBM to solve Problem (4)
        starting from each point $(\boldsymbol{x}, y) \in S_1$;
    END SOLVING AUXILIARY CLR PROBLEM
    SOLVING CLR PROBLEM
      For each $(\bar{\boldsymbol{x}}, \bar{y}) \in S_2$ apply LMBM to solve Problem (2)
        starting from the point $(\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_{l-1}, y_{l-1}, \bar{\boldsymbol{x}}, \bar{y})$ and find
        a solution $(\hat{\boldsymbol{x}}_1, \hat{y}_1, \ldots, \hat{\boldsymbol{x}}_l, \hat{y}_l)$;
      Denote by $S_3 \subset \mathbb{R}^{(n+1)l}$ a set of all such solutions;
    END SOLVING CLR PROBLEM
    SOLUTION TO THE $l$ -CLR PROBLEM
      Compute $f_l^{min} = \min\{f_l(\hat{\boldsymbol{x}}_1, \hat{y}_1, \ldots, \hat{\boldsymbol{x}}_l, \hat{y}_l) \mid (\hat{\boldsymbol{x}}_1, \hat{y}_1, \ldots, \hat{\boldsymbol{x}}_l, \hat{y}_l) \in S_3\}$
        and the collection of linear functions $(\bar{\boldsymbol{x}}_1, \bar{y}_1, \ldots, \bar{\boldsymbol{x}}_l, \bar{y}_l)$
        such that $f_l(\bar{\boldsymbol{x}}_1, \bar{y}_1, \ldots, \bar{\boldsymbol{x}}_l, \bar{y}_l) = f_l^{min}$;
      Set $\boldsymbol{x}_j = \bar{\boldsymbol{x}}_j, y_j = \bar{y}_j$, $j = 1, \ldots, l$ as a solution to the $l$-CLR
        problem;
    END SOLUTION TO THE $l$ -CLR PROBLEM
  END WHILE
  RETURN the solution to the $k$-CLR problem;
END PROGRAM Incremental Algorithm
```

In addition to the k-CLR problem, LMBM-CLR solves also all intermediate l-CLR problems, where $l = 1, \ldots, k-1$.

4 Numerical Experiments

The proposed algorithm was tested using some data sets for regression analysis. First, we apply the algorithm to solve CLR problems in small real world data sets with known solutions. Then we solve CLR problems in large real world data sets. We compare the proposed algorithm with the `NOBIA-CLR` — Nonsmooth Optimization Based Incremental Algorithm for Clusterwise Linear Regression [3] — and the well-known multistart Späth's-algorithm `MS-Späth` [27, 28] using results from these data sets.

`LMBM-CLR` is implemented in Fortran 95 while `NOBIA-CLR` and `MS-Späth` are implemented in Fortran 77. All the software are compiled using `gfortran`, the GNU Fortran compiler. The experiments were performed on MacBookAir (OS El Capitan 10.11.3) with Intel® Core™ i5, 1.6 GHz and RAM 4 GB.

`LMBM-CLR` and `NOBIA-CLR` methods use the incremental approach to solve CLR problems globally while `MS-Späth` uses the simple randomized multistart scheme for starting points. Thus, `MS-Späth` does not give any intermediate results. For comparison purposes, we made different runs for different numbers of linear regression functions.

The following notations are used to present computational results:

- m is the number of observations (data points);
- n is the number of input variables;
- k is the number of linear regression functions (or clusters);
- f_{best} is the best known value of the function f_k;
- E_A is the error in $\%$;
- nf_a is the number of auxiliary regression error function (3) evaluations.
- nf_r is the number of regression error function (1) evaluations.
- N_{reg} is the number of linear regression problems (2) solved; and
- *cpu* is the used cpu time in seconds.

The error E_A is computed as

$$E_A = \frac{(f_A - f_{best})}{f_{best} + 1} \times 100, \tag{5}$$

where f_A is the value of the function f_k obtained by an algorithm A. $E_A = 0$ implies that an algorithm finds the best known solution. We have used the f_{best} value given in [3] (for those data sets that were used also in [3]) unless we got better value in our experiments.

4.1 Results on small data sets with known solution

In this subsection we present results with 20 small size data sets available from [7]. The brief description of the data sets is given in Table 1. Computational results for these data sets and the values of global solutions are given in Tables 2 and 3. We include here only the error value E_A for each algorithm, since all the algorithms found the solution immediately. With `MS-Späth` we started the computations from 1000 randomly chosen initial points. Tables 2 and 3 contain results with two and three regression functions, respectively. We used only a subset of the data sets given in Table 1 to compute three regression functions since only for these data sets the values at global solutions are known.

Table 1: The brief description of small data sets

Data set		m	n	Data set		m	n
1	Acorn	39	3	11	Extroversion	40	3
2	Brinks	47	4	12	House Prices	40	1
3	Car Fuel	82	1	13	Mercury Bass	53	4
4	CEO Salaries	59	1	14	Mortality	58	3
5	Check Off	56	4	15	Nuclear Plants	32	3
6	Cheese Taste	30	3	16	Polishing Times	59	2
7	Crime Rates	47	5	17	Public Expenditure	48	4
8	Diabetes	47	2	18	Smoking Cancer	44	1
9	Electricity	50	3	19	Temperatures	56	2
10	Enrollment	29	4	20	Votes	50	1

We conclude from these results that the accuracy of the new algorithm `LMBM-CLR` was similar to that of `NOBIA-CLR`. With two regression functions these solvers found the near global solution ($E_{\texttt{LMBM-CLR}} \leq 1.84$ and $E_{\texttt{NOBIA-CLR}} \leq 2.51$) in all but three data sets. The accuracy of `MS-Späth` was a little bit worse: it solved 15 problems with error less or equal to 2.82. All the solvers totally failed to solve the Brinks data set (data set 2, see Table 2).

Obviously, reaching the global solution was much more difficult task with three regression functions (see Table 3). Here, the accuracy of `LMBM-CLR` was again similar to that of `NOBIA-CLR`. `LMBM-CLR` solved six problems (out of 11) to near global solution ($E_{\texttt{LMBM-CLR}} \leq 2.90$) while with `NOBIA-CLR` the number of accurate solutions were five but all of them were obtained with $E_{\texttt{NOBIA-CLR}} = 0.00$. Again `MS-Späth` was the most inaccurate one: it solved only three problems with $E_{\texttt{MS-Späth}} \leq 3.95$.

4.2 Results on large real data sets

In this subsection we report results on nine large real world data sets for regression. The brief description of these data sets is given in Table 4. Their detailed descriptions

Table 2: Results for small data sets with two regression functions

Data	f_{best}	$E_{\texttt{LMBM-CLR}}$	$E_{\texttt{NOBIA-CLR}}$	$E_{\texttt{MS-Späth}}$
1	223817911.1	0.00	0.00	7.52
2	382570037.000	114658.82	99961.37	103199.50
3	335.95050870	0.00	0.00	1.80
4	768960.336	0.00	0.00	0.00
5	1169.2983490	0.00	0.00	0.00
6	503.7991483	0.00	0.00	2.62
7	3329.26314	27.00	49.56	53.15
8	3.640144568	0.00	0.00	0.00
9	284.5280943	0.01	2.51	4.72
10	969887.3	0.00	0.00	29.15
11	1478.422187	1.84	0.00	1.84
12	51655.04437	0.00	0.00	0.00
13	0.550461693	0.00	0.00	0.00
14	0.000000000	0.00	0.00	0.00
15	112810.03462	1.46	0.00	2.82
16	5315.41062	0.04	0.00	0.04
17	13660.93458	0.71	0.00	0.71
18	140.3610520	0.00	0.00	0.00
19	428.3865333	24.71	24.71	0.00
20	189.3029607	0.00	0.00	0.00

Table 3: Results for small data sets with three regression functions

Data	f_{best}	$E_{\texttt{LMBM-CLR}}$	$E_{\texttt{NOBIA-CLR}}$	$E_{\texttt{MS-Späth}}$
1	70493712.1	14.82	37.57	24.31
6	95.0925567	89.48	46.60	120.64
8	1.332098236	0.33	0.00	12.26
10	97879.0953	261.80	98.12	184.50
11	415.7273293	1.85	15.17	9.58
12	19540.35008	2.90	0.00	11.68
15	10714.82356	33.71	46.34	22.38
16	1900.03938	0.00	0.00	13.93
18	57.7651194	0.00	0.00	0.00
19	145.6685991	26.83	7.85	3.95
20	91.1856555	0.00	0.00	0.00

can be found in [23] and references given in Table 4. The number of input variables in these data sets ranges from 4 to 280 and the number of data points from 1030 to 52397. All input variables are numeric and data sets do not contain missing values. We compute up to ten linear regression functions in each data set. Results are presented in Tables 5–13 and Figures 1–9.

To obtain comparable results, the numbers of different starting points for `MS-Späth` was always kept big enough, but we limited the computational time of the method to be twice of that used by `LMBM-CLR`. We also stopped the run if the wall clock time was more than 24 hours without any progress. That is, if after 24 hours `MS-Späth` was still computing the solution from the first starting point.

Table 4: The brief description of large data sets

Data set	m	n	Reference
Concrete compressive strength	1030	8	[33]
Airfoil self-noise	1503	5	[23]
Red wine quality	1599	11	[10]
White wine quality	4898	11	[10]
Insurance company benchmark (COIL 2000)*	5822	85	[30]
Combined cycle power plant	9568	4	[29, 21]
Online News Popularity	39644	58	[13]
Physicochemical properties of protein tertiary structure	45730	9	[23]
BlogFeedback*	52397	280	[5]

* training data set.

Let us first examine data sets with less than 6000 data points (5 – 85 input variables). That is, Concrete compressive strength, Airfoil self-noise, Red wine quality, White wine quality, and Insurance company benchmark data sets. Note that in Red wine quality, White wine quality, and Insurance company benchmark data sets the values of the regression error functions were equal to zero with 6, 7, and 2 linear regression functions, respectively, and we stopped computing more regression functions in these cases.

As with small data sets, `LMBM-CLR` and `NOBIA-CLR` behaved quite similarly: both had some difficulties in Concrete compressive strength data set with larger number of regression functions, nevertheless not as bad as `MS-Späth` (see Table 5); `LMBM-CLR` and `NOBIA-CLR` did not find the best known solution in Red wine quality data set with five regression functions, on the other hand, `MS-Späth` found it only with two regression functions (see Table 7); and in Airfoil self-noise data set `LMBM-CLR` failed to find the best known solution with seven regression function while `NOBIA-CLR` failed with ten regression functions (see Table 6). Again `MS-Späth` did not find the best known solution with any numbers of regression functions. Moreover, MS-Späth failed to find accurate solutions in White vine quality data set but with two regression functions and in Insurance company benchmark data set (see Tables 8 and 9). Besides above the solvers found at least near the best known solution with $E_A \leq 0.19$.

MS-Späth algorithm used less regression function calculations than the other two algorithms when there were only few regression functions involved. However, with larger number of regression functions MS-Späth algorithm needed significantly more computations. Note that the procedure used in MS-Späth is completely different from that of the other two algorithms and, although, the procedures for finding a global solution in LMBM-CLR and NOBIA-CLR are quite similar, their internal implementation is somewhat different. Thus, a sole number of regression function calculations does not give a right impression of the computational burden. Indeed, LMBM-CLR needed to solve more regression problems than NOBIA-CLR but to do so, it used significantly less function evaluations, especially, evaluations of the computationally more expensive regression error function (1). As a result LMBM-CLR was about five times faster than NOBIA-CLR. In Figures 1 – 5 we have compared the used CPU times of LMBM-CLR and NOBIA-CLR when the numbers of linear regression functions are increased. As already said MS-Späth always uses (at least) twice the computation time of LMBM-CLR and thus, we omitted it from these figures. These figures show that the computational time of LMBM-CLR increased clearly less with the numbers of regression functions.

Second, we consider data sets with less than 50000 data points (4 – 58 input variables): that is Combined cycle power plant, Online News Popularity, and Physicochemical properties of protein tertiary structure data sets (see Tables 10 – 12). As before LMBM-CLR and NOBIA-CLR behaved quite similarly and MS-Späth was very inaccurate. In fact, we could say that MS-Späth did not solve these problems. Here, LMBM-CLR was clearly faster than NOBIA-CLR due to fact that LMBM-CLR solved less regression problems than NOBIA-CLR and also significantly less function evaluations, especially, evaluations of the computationally more expensive regression error function (1). The comparisons of the used CPU times of LMBM-CLR and NOBIA-CLR are given in Figures 6 – 8. Again, these figures show that the computational time of LMBM-CLR was significantly smaller and increased clearly less with the number of regression functions.

Table 5: Results for Concrete compressive strength data set.

k	f_{best}	`LMBM-CLR`					`NOBIA-CLR`					`MS-Späth`		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	29518.84	0.01	11157	696	962	0.53	0.00	85750	15505	937	3.69	0.04	262	1.11
3	12748.52	0.02	26726	8773	2034	1.73	0.00	156528	74540	1880	11.06	28.20	1356	3.52
5	4474.77	4.61	61079	38605	4958	6.44	0.00	272625	210200	4166	31.59	50.83	8045	13.05
7	1981.18	11.24	81608	65378	7734	11.58	9.60	388375	371598	7419	61.17	127.77	20790	23.60
10	870.06	5.96	102458	93439	11497	18.41	6.48	475473	494412	10443	91.62	303.90	44110	37.06

Table 6: Results for Airfoil self-noise data set

k	f_{best}	`LMBM-CLR`					`NOBIA-CLR`					`MS-Späth`		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	11616.29	0.01	14677	1261	1426	0.64	0.00	64339	5655	1073	2.36	15.50	284	1.42
3	5834.05	0.00	28281	2642	2678	1.20	0.00	121127	14554	2028	4.64	23.50	879	2.66
5	2359.72	0.19	55482	8625	5372	2.67	0.00	221177	44635	3921	10.67	16.60	3415	5.45
7	1217.16	3.29	85366	22204	9192	5.62	0.00	332120	112489	6737	24.90	51.82	10682	11.95
10	532.25	0.00	118136	45082	15178	11.66	4.14	474623	186397	10661	46.11	190.10	27790	23.65

Table 7: Results for Red wine quality data sets

k	f_{best}	`LMBM-CLR`					`NOBIA-CLR`					`MS-Späth`		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	234.93	0.05	15044	3196	1298	1.68	0.00	87793	21734	1041	7.96	0.02	552	3.76
3	90.95	0.00	28653	10588	2284	3.61	0.00	123225	49995	1570	14.21	42.97	1719	7.70
5	5.58	10.65	28913	10828	2380	3.70	10.65	131659	52932	1731	15.19	137.10	6035	16.93
6	0.00	0.00	29128	12261	2468	4.09	0.00	136082	57545	1777	16.66	427.88	7668	28.56

Table 8: Results for White wine quality data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	1051.33	0.00	8876	702	577	2.15	0.00	28051	12248	291	9.43	0.69	154	4.75
3	357.45	0.00	11402	1810	819	3.15	0.00	40821	20130	472	15.46	94.24	363	7.50
5	18.99	0.00	11727	1900	1038	3.48	0.00	45655	21579	609	17.26	1798.42	575	9.44
6	4.26	0.00	11742	1948	1058	3.60	0.00	46056	22155	629	17.91	1284.58	702	8.07
7	0.00	0.00	11757	2235	1103	3.94	0.00	46953	23136	644	19.12	182.47	1085	11.05

Table 9: Results for Insurance company benchmark data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	0.00	0.00	9916	38049	931	193.40	0.00	21311	170807	382	737.71	126.89	276	415.70

Table 10: Results for Combined cycle power plant data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	71058.91	0.00	3195	72	238	0.93	0.00	17109	849	335	4.39	17.90	120	2.81
3	40414.89	0.01	6540	155	463	1.88	0.00	34006	1488	640	8.70	29.90	285	5.00
5	18288.70	0.00	11214	2360	852	4.69	0.00	61241	23139	1232	29.85	24.33	1125	13.61
7	10121.22	0.22	17213	3964	1435	7.68	0.00	84246	31819	1756	43.55	31.33	2380	22.29
10	5276.79	1.03	19271	5819	1747	11.17	0.00	94806	44502	2137	62.81	42.52	4630	31.62

Table 11: Results for Online news popularity data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	1346500473599.88	9.47	1864	458	11	42.93	0.00	2191	784	14	57.40	968535.35	2000	7776.26
3	494250920008.32	7.39	2157	4088	16	146.48	0.00	20351	42344	184	1103.77	131578.87	3000	8644.76
5	129195562671.60	0.00	2937	12824	29	436.24	2.49	27003	79614	272	2252.11	74334.11	5000	10876.94
7	52957003192.86	3.79	3488	20746	46	778.84	0.00	36356	100514	354	3133.25	5400.67	7000	10553.54
10	20598655690.93	4.03	4541	33318	79	1471.63	0.00	39939	118716	429	4080.64	17532.86	10000	11140.76

Table 12: Results for Physicochemical properties of protein data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	214398.13	0.00	98	134	5	12.00	0.00	3269	2633	26	38.94	$7.29 \cdot 10^{12}$	60	24.95
3	106427.64	0.00	209	302	10	23.98	0.00	6090	3261	43	71.54	$14.69 \cdot 10^{12}$	162	49.90
5	41640.04	0.18	430	1042	23	52.99	0.00	9198	7954	91	150.59	$80.25 \cdot 10^{12}$	465	109.72
7	22184.81	0.08	658	2451	40	87.22	0.00	11983	14141	155	250.15	$70.40 \cdot 10^{12}$	1295	183.26
10	11387.88	0.00	933	3720	73	133.67	0.12	15204	21856	246	407.20	$137.26 \cdot 10^{12}$	2500	272.65

Table 13: Results for BlogFeedback data set.

k	f_{best}	LMBM-CLR					NOBIA-CLR					MS-Späth		
		E_A	nf_a	nf_r	N_{reg}	cpu	E_A	nf_a	nf_r	N_{reg}	cpu	E_A	N_{reg}	cpu
2	10071339.86	6.33	5001	4167	5	1400.76	18.28	1094	4693	5	985.55	0.00	170	28132.68
3	3574337.09	0.00	8004	9167	10	2581.12	16.53	2257	9258	10	1835.74	152.32	255	30887.51
5	1015242.00	0.00	13851	18343	23	5226.69	4.94	4680	25420	23	5614.43	276.44	740	57301.91
7	314172.23	0.00	19122	26566	40	8114.45	22.14	7047	36642	40	9324.52	2458.66	882	55978.93
10	96360.02	0.00	29133	32104	73	11073.47	20.64	9303	57562	73	17901.96	*		

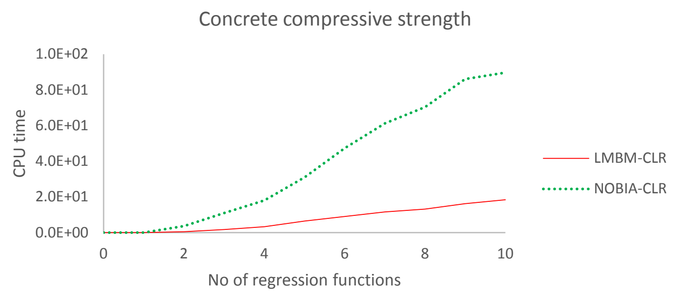

Figure 1: Concrete compressive strength: used CPU time vs. numbers of regression functions.

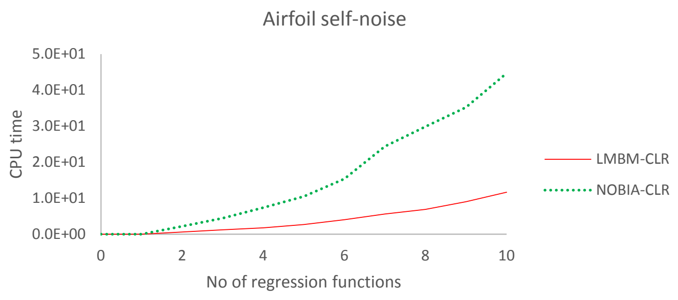

Figure 2: Airfoil self-noise: used CPU time vs. number of regression functions.

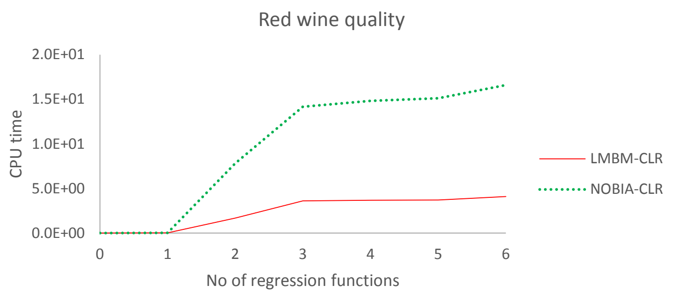

Figure 3: Red wine quality: used CPU time vs. number of regression functions.

All the results of LMBM−CLR and NOBIA−CLR are at least near the best known solution in Combined cycle power plant and Physicochemical properties of protein tertiary structure data sets ($E_A \leq 1.3$, see Tables 10 and 12). However, in Online News Popularity data set LMBM−CLR had problems with accuracy: it found the best known solution only with five regression functions, otherwise $3.79 \leq E_{\texttt{LMBM-CLR}} \leq 9.47$ (see Table 11).

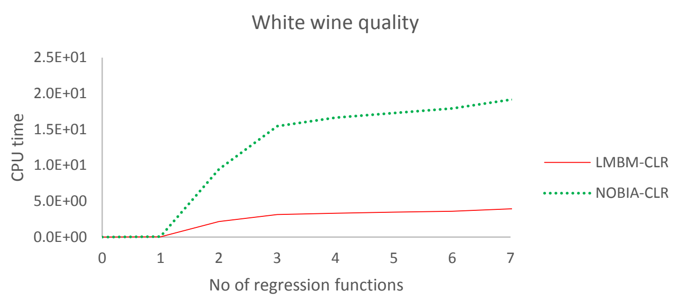

Figure 4: White wine quality: used CPU time vs. number of regression functions.

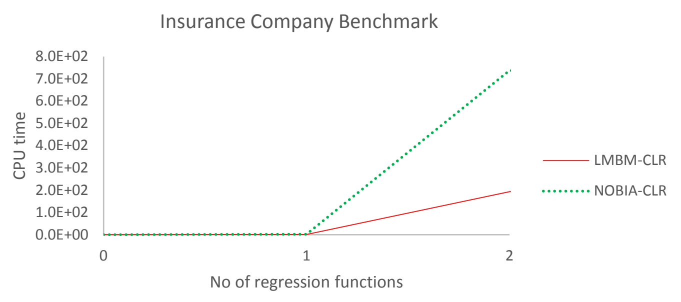

Figure 5: Insurance company benchmark: used CPU time vs. number of regression functions.

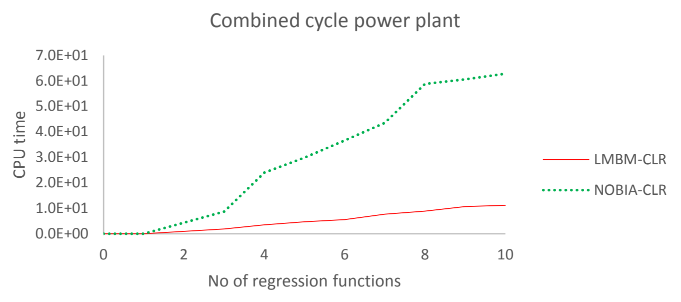

Figure 6: Combined cycle power plant: used CPU time vs. number of regression functions.

Finally, in BlockFeedback data set which has both very large number of data points (52397) and a large number of input variables (280) we obtained a little bit different results (see Table 13 and Figure 9). With two regression functions `MS-Späth` found the smallest minimum. Nevertheless, with larger number of regression functions `MS-Späth` failed to find even near best known solutions. In addition, with less than five regression functions `LMBM-CLR` used more CPU time than `NOBIA-CLR`. However, `LMBM-CLR` found smaller function values in all cases and with ten re-

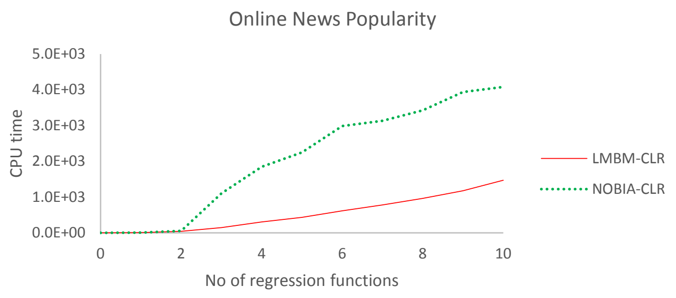

Figure 7: Online news popularity: used CPU time vs. number of regression functions.

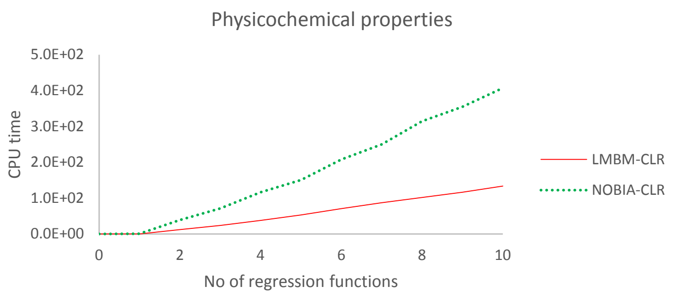

Figure 8: Physicochemical properties of protein: used CPU time vs. number of regression functions.

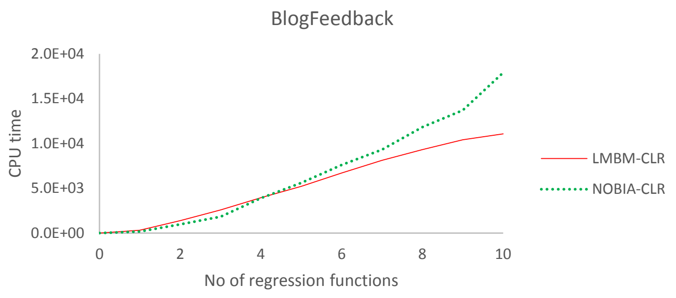

Figure 9: BlogFeedback: used CPU time vs. number of regression functions.

gression functions `LMBM-CLR` was also significantly faster than `NOBIA-CLR` . `LMBM-CLR` and `NOBIA-CLR` solved an equal number of regression problems. While `NOBIA-CLR` usually used less auxiliary error function evaluations `LMBM-CLR` succeed solving problems with less regression error function evaluations.

Note that with Online news popularity and BlockFeedback data sets `MS-Späth` computed solutions from only one starting point before the time limit was exceeded. In addition, we did not obtain any result from `MS-Späth` in BlockFeedback data set with ten regression function within 24 hours.

5 Conclusions

In this paper a new LMBM-CLR -method for solving the clusterwise linear regression problems is introduced. Here the clusterwise linear regression problem is formulated as a nonsmooth nonconvex optimization problem. In addition, an auxiliary clusterwise linear regression problem is introduced to find initial solutions for the clusterwise linear regression problem.

The LMBM-CLR -method consist of two different algorithms: an incremental algorithm is used to solve clusterwise linear regression problems globally and at each iteration of this algorithm the LMBM algorithm is used to minimize both the clusterwise linear regression error function and the auxiliary clusterwise linear regression error function with different starting points.

The new LMBM-CLR -method was tested using both small data sets with known solutions and large real world data sets with the number of data points ranging from thousands to tens of thousands. In small data sets the difference in computational times was insignificant. The accuracy of the new method was similar to that of the NOBIA-CLR and better than Späth's algorithm. In large data sets the new method LMBM-CLR was significantly faster and at least as accurate as the other methods tested. Numerical results demonstrate that the LMBM-CLR is especially efficient for solving CLR problems in data sets with large number of input variables.

Acknowledgments

The work was financially supported by the Academy of Finland (Project No. 289500) and Australian Research Counsil's Discovery Projects funding scheme (Project No. DP140103213). The work was made while the corresponding author was visiting in Faculty of Science and Technology, Federation University Australia.

References

[1] BAGIROV, A., UGON, J., AND MIRZAYEVA, H. Nonsmooth nonconvex optimization approach to clusterwise linear regression problems. *European Journal of Operational Research 229*, 1 (2013), 132–142.

[2] BAGIROV, A., UGON, J., AND MIRZAYEVA, H. An algorithm for clusterwise linear regression based on smoothing techniques. *Optimization Letters 9*, 2 (2015), 375–390.

[3] BAGIROV, A., UGON, J., AND MIRZAYEVA, H. Nonsmooth optimization algorithm for solving clusterwise linear regression problems. *Journal of Optimization Theory and Applications 164*, 3 (2015), 755–780.

[4] BIHAIN, A. Optimization of upper semidifferentiable functions. *Journal of Optimization Theory and Applications 4* (1984), 545–568.

[5] BUZA, K. Feedback prediction for blogs. In Data Analysis, Machine Learning and Knowledge Discovery (pp. 145-152). Springer International Publishing., 2014. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[6] BYRD, R. H., NOCEDAL, J., AND SCHNABEL, R. B. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming 63* (1994), 129–156.

[7] CARBONNEAU, R. Clusterwise regression datasets. <URL: `http://rcarbonneau.com/ClusterwiseRegressionDatasets.htm/`> (June 11th, 2016), 2015.

[8] CARBONNEAU, R., CAPOROSSI, G., AND HANSEN, P. Globally optimal clusterwise regression by mixed logical-quadratic programming. *European Journal of Operational Research 212* (2011), 213–222.

[9] CARBONNEAU, R., CAPOROSSI, G., AND HANSEN, P. Extensions to the repetitive branch-and-bound algorithm for globally-optimal clusterwise regression. *Computers and Operations Research 39*, 11 (2012), 2748–2762.

[10] CORTEZ, P., CERDEIRA, A., ALMEIDA, F., MATOS, T., AND REIS, J. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems, Elsevier 47*, 4 (2009), 547–553. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[11] DESARBO, W., AND CRON, W. A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification 5*, 2 (1988), 249–282.

[12] DESARBO, W., OLIVER, R., AND RANGASWAMY, A. A simulated annealing methodology for clusterwise linear regression. *Psychometrika 54*, 4 (1989), 707–736.

[13] FERNANDES, K., VINAGRE, P., AND CORTEZ, P. A proactive intelligent decision support system for predicting the popularity of online news. Proceedings of the 17th EPIA 2015 — Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal., 2015. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[14] GAFFNEY, S., AND SMYTH, P. Trajectory clustering using mixtures of regression models. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining* (1999), S. Chaudhuri and D. Madigan, Eds., New York, pp. 63–72.

[15] GARCÌA-ESCUDERO, L., GORDALIZA, A., MAYO-ISCAR, A., AND SAN MARTIN, R. Robust clusterwise linear regression through trimming. *Computational Statistics and Data Analysis 54* (2010), 3057–3069.

[16] HAARALA, M. *Large-Scale Nonsmooth Optimization: Variable Metric Bundle Method with Limited Memory*. PhD thesis, University of Jyväskylä, Department of Mathematical Information Technology, 2004.

[17] HAARALA, M., MIETTINEN, K., AND MÄKELÄ, M. M. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software 19*, 6 (2004), 673–692.

[18] HAARALA, N., MIETTINEN, K., AND MÄKELÄ, M. M. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming 109*, 1 (2007), 181–205.

[19] HIRIART-URRUTY, J.-B., AND LEMARÉCHAL, C. *Convex Analysis and Minimization Algorithms II*. Springer-Verlag, Berlin, 1993.

[20] KARMITSA, N., MÄKELÄ, M. M., AND ALI, M. M. Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. *Applied Mathematics and Computation 198*, 1 (2008), 382–400.

[21] KAYA, H., TÜFEKCI, P., AND GÜRGEN, S. F. 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., 2012. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[22] KIWIEL, K. C. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics 1133. Springer-Verlag, Berlin, 1985.

[23] LICHMAN, M. UCI machine learning repository. Available in web page <URL: `http://archive.ics.uci.edu/ml`>, University of California, Irvine, School of Information and Computer Sciences, 2013. (April 8th, 2016).

[24] LUKŠAN, L., AND VLČEK, J. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications 102*, 3 (1999), 593–613.

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

[26] PREDA, C., AND SAPORTA, G. Clusterwise pls regression on a stochastic process. *Computational Statistics & Data Analysis 49* (2005), 99–108.

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

[28] SPÄTH, H. Algorithm 48: A fast algorithm for clusterwise linear regression. *Computing 29* (1981), 175–181.

[29] TÜFEKCI, P. 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* (2014), 126–140. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[30] VAN DER PUTTEN, P., AND VAN SOMEREN, M. E. Coil challenge 2000: The insurance company case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000., 2000. Data set available in UCI machine learning repository <URL: `http://archive.ics.uci.edu/ml`> (June 11th, 2016).

[31] VLČEK, J., AND LUKŠAN, L. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications 111*, 2 (2001), 407–430.

[32] WEDEL, M., AND KISTEMAKER, C. Consumer benefit segmentation using clusterwise linear regression. *International Journal of Research in Marketing 6*, 1 (1989), 45–59.

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

ISBN 978-952-12-3488-0
ISSN 1239-1891