Ann Oper Res (2017) 249:273–300
DOI 10.1007/s10479-016-2333-y

PARDALOS60

DCA based algorithms for feature selection in multi-class support vector machine

Hoai An Le Thi[1,2] · **Manh Cuong Nguyen**[2]

Published online: 4 October 2016

Abstract This paper addresses the problem of feature selection for Multi-class Support Vector Machines. Two models involving the ℓ_0 (the zero norm) and the ℓ_2–ℓ_0 regularizations are considered for which two continuous approaches based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) are investigated. The first is DC approximation via several sparse inducing functions and the second is an exact reformulation approach using penalty techniques. Twelve versions of DCA based algorithms are developed on which empirical computational experiments are fully performed. Numerical results on real-world datasets show the efficiency and the superiority of our methods versus one of the best standard algorithms on both feature selection and classification.

Keywords Feature selection · MSVM · DC programming · DCA · DC approximation · Exact penalty

1 Introduction

One of challenges of Machine Learning is the handling of the input datasets with very large number of features. The so called *feature selection* has been widely studied to address this challenge. The goals are to remove the irrelevant and redundant features, reduce store space and execution time, and avoid the course of dimensionality to improve the prediction performance (Le Thi et al. 2008b).

✉ Hoai An Le Thi
lethihoaian@tdt.edu.vn

Manh Cuong Nguyen
manh-cuong.nguyen@univ-lorraine.fr

1 Department for Management of Science and Technology Development and Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam

2 Laboratory of Theoretical and Applied Computer Science LITA EA 3097, University of Lorraine, Ile du Saulcy, 57045 Metz, France

The research on feature-selection methods is very active in recent years, and an excellent review can be found in the book by Guyon et al. (2006). Generally speaking, feature selection can be classified into three categories: filter approaches, wrapper approaches, and embedded approaches. Wrapper methods exploit a machine learning algorithm to evaluate the usefulness of features. Filter methods rank the features according to some discrimination measure and select features having higher ranks without using any learning algorithm (it utilizes the underlying characteristics of the training data to evaluate the relevance of the features or feature set by some independent measures such as distance measure, correlation measures, consistency measures Chen et al. 2006). The wrapper approach is generally considered to produce better feature subsets but runs much more slowly than a filter. In contrast to the filter and wrapper approaches, the embedded approach of feature selection does not separate the learning from the feature selection part. It integrates the selection of features in the model building. For instance, for feature selection in classification, an embedded method uses a machine learning algorithm to search a classifier that uses as few features as possible while a filter method selects the features by optimizing a measure criterion and then finds a classifier defined on the selected features.

Feature selection is often applied to high-dimensional data prior to classification learning. In this paper, we are interested in the feature selection task for Multi-class Support Vector Machine (MSVM) in the framework of embedded approaches. The objective is to simultaneously select a subset of features (representative features) and construct a good classifier, i.e. we search a sparse MSVM. Whereas most feature selection methods were initially developed for binary-Support Vector Machine (SVM) classification (see e.g. Bradley and Mangasarian 1998; Fan and Li 2001; Hermes and Buhmann 2000; Hui 2006; Le Thi et al. 2008a, b; Neumann et al. 2005; Rakotomamonjy 2003; Wang et al. 2007), several extensions to feature selection for MSVM are recently investigated (see e.g. Cai et al. 2011; Chapelle 2008; Chen et al. 2006; Deng et al. 2013; Duan et al. 2005; Huang et al. 2013; Hsu and Lin 2002; Lee et al. 2006, 2004; Li et al. 2004; Wang and Shen 2003; Weston and Watkins 1999; Weston et al. 2003; Wu et al. 2007; Zhang et al. 2008; Zou 2006; Zhou and Tuck 2007; Zhou et al. 2010). However, an extension from the binary case to the multi-category case (Q classes) is not trivial. Indeed, as will be seen in the next section, the decision rule in MSVM has to be determined from Q decision functions (but not one decision function as in SVM) and each selected feature should be associated with Q coefficients in Q decision functions.

For the feature selection purpose, we use a natural concept dealing with sparsity, that is the zero norm (denoted ℓ_0 or $\|.\|_0$). The zero norm of a vector is defined as the number of its nonzero components. The function ℓ_0, apparently very simple, is lower-semicontinuous on $\mathbb{R}^n$, but its discontinuity at the origin makes nonconvex programs involving $\|.\|_0$ challenging. During the last two decades, research is very active in optimization models and methods involving the zero norm. Works can be divided into three categories according to the way to treat the zero norm: convex approximation (the ℓ_0-norm is replaced by a convex function, for instance the ℓ_1-norm Tibshirani 1996 or the conjugate function Pham Dinh and Le Thi 2014), nonconvex approximation (a continuous nonconvex function is used instead to the ℓ_0-norm, usual sparse inducing functions are introduced in Bradley and Mangasarian (1998), Fan and Li (2001), Le Thi (2012), Le Thi et al. (2015b), Peleg and Meir (2008), Weston et al. (2003)), and nonconvex exact reformulation (with the binary variables $u_i = 0$ if $w_i = 0$ and $u_i = 1$ otherwise, the original problem is formulated as a combinatorial optimization problem which is equivalently reformulated as a continuous nonconvex program via exact penalty techniques (Le Thi et al. 2015a)). An extensive overview of these approaches can be found in Le Thi et al. (2015b). When the objective function (besides the ℓ_0-term) is convex,

convex approximation techniques result in a convex optimization problem which is so far "easy" to solve.

Whilst nonconvex approximation approaches have been widely used for feature selection in SVM, most of works on feature selection in MSVM are based on convex approximations, especially on ℓ_1 regularization. In this work, we study two models of sparse MSVM using the ℓ_0 and/or the ℓ_2–ℓ_0 regularization. Our motivation to consider the ℓ_2–ℓ_0 regularization is that it can reduce overfitting. Due to the ℓ_0 term, the resulting problems are nonsmooth and nonconvex. We tackle these problems by the two nonconvex approaches, both are based on Difference of Convex functions (DC) programming and DC Algorithms (DCA), powerful tools in the nonconvex programming framework which were introduced by Pham Dinh Tao in their preliminary form in 1985 and have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao and become now classic and increasingly popular (see, e.g. Le Thi and Pham Dinh 2005; Le Thi 2005; Pham Dinh and Le Thi 1997, 1998, and references therein).

Our motivating arguments to use the ℓ_0-norm are multiple. Firstly, even though using the ℓ_1 is the simplest way to deal with the sparsity, the ℓ_1 can encourage the sparsity in only some cases with restrictive assumptions (see Gribonval and Nielsen 2003). In particular, for feature selection purpose, the ℓ_1 penalty has been shown to be, in certain cases, inconsistent and biased (Zou 2006). Secondly, the ℓ_0-norm is the most natural and suitable concept for modelling the sparsity, and nonconvex approximations of the ℓ_0-norm are, in general, deeper than the ℓ_1-norm, and can then produce better sparsity. Especially, for feature selection in SVM, solutions of the ℓ_0 -norm penalty problem have been shown to be much sparser than those of ℓ_1-norm approach in several previous works (see e.g. Le Thi et al. 2008a, 2015a, b; Ong and Le Thi 2013). Thirdly, although we are faced with nonconvex problems, the power of DCA can be exploited to efficiently solve these hard problems, knowing that DCA has been successfully developed in a variety of works in Machine Learning (see e.g. Collobert et al. 2006; Krause and Singer 2004; Le Thi et al. 2006, 2007, 2008a, b, 2015a, b; Le Thi and Phan 2016a, b; Liu et al. 2005; Liu and Shen 2006; Ronan et al. 2006 and the list of reference in Le Thi (2005)), in particular to feature selection in SVM (Le Thi et al. 2008a, b, 2015a, b; Neumann et al. 2005; Ong and Le Thi 2013).

In our first approach, the ℓ_0-norm is approximated by a DC function that leads to a DC program for which a DCA scheme is investigated. This general DCA scheme is developed to various sparse inducing DC approximation functions: the piecewise exponential function (Bradley and Mangasarian 1998), the SCAD penalty function (Fan and Li 2001), the logarithm function (Weston et al. 2003), the capped-ℓ_1 function (Peleg and Meir 2008) and the piecewise linear function recently proposed in Le Thi (2012). In the second approach, the original problem is equivalently reformulated, via an exact penalty technique in DC programming (Le Thi et al. 2012), as a DC program. Hence, using a unified DC programming framework, we unify all solution methods into DCA, and then convergence properties of our algorithms are guaranteed thanks to general convergence results of the generic DCA scheme. Specific convergence properties of each DCA secheme are also studied. We perform empirical comparative numerical experiments of 12 versions of DCA based algorithms, with various approximate functions as well as with the exact continuous reformulation. We are interested in several questions from both algorithmic and numerical points of view: what is better between the ℓ_0 or ℓ_2–ℓ_0 regularization? What might be the approach advised—the nonconvex approximation or the nonconvex exact penalty reformulation? And in the nonconvex approximation approaches, what is the best approximation among several sparse inducing functions? Such questions are useful for researchers in the choice of the algorithm to be applied to their problems among various versions offered in this paper.

The remainder of the paper is organized as follows. Section 2 contains the introduction of two models of sparse MSVM using ℓ_0 and ℓ_2–ℓ_0 regularizations, followed by a brief presentation of DC Programming and DCA. The approximation approach is presented in Sect. 3 while the exact penalty approach is developed in Sect. 4. Computational experiments are reported in Sect. 5 and finally Sect. 6 concludes the paper.

2 Models and methodology

2.1 Sparse MSVM models

For beginning, let us introduce the model of MSVM proposed by Weston and Watkins (1999), a direct approach (without using binary-SVM) for learning multiclass, known to be appropriate to capture correlations between the different classes, which can be described as follows.

Let $\mathcal{X}$ be a set of vectors in $\mathbb{R}^d$ and $\mathcal{Y} = \{1, \ldots, Q\}$ be a set of class labels. Given a training dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \in \mathbb{R}^{n\times(d+1)}$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}, i = \{1, \ldots, n\}$. The task is to learn a classification rule $f : \mathcal{X} \mapsto \mathcal{Y}$ that maps an element x to a class label $y \in \mathcal{Y}$.

In a more natural way than the classical SVM based approaches for multi-classification, Weston and Watkins (1999) proposed to construct a piecewise linear separation that gives the decision function:

$$f(x) = \arg\max_{1\le i\le Q} f_i(x), \tag{1}$$

where f_i stands for the hyperplane $f_i(x) = \langle w_i, x\rangle + b_i$, with $w_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}, i = 1, \ldots, Q$. Let $w = \left(w_1, w_2, \ldots, w_Q\right)$ be the vector in $\mathbb{R}^{Q\times d}$ and let $\boldsymbol{b} = (b_i)_{i=1}^{Q} \in \mathbb{R}^Q$. Then the MSVM model given in Weston and Watkins (1999), the first "all-together" implementation of multi-class SVM, is a single optimization problem of the form:

$$\min\left\{ C\sum_{i=1}^{n}\sum_{k\neq y_i} \xi_{ik} + \sum_{k=1}^{Q} \|w_k\|_2^2 : (w, b, \xi) \in \Omega \right\}, \tag{2}$$

where

$$\Omega = \left\{ \begin{array}{l} (w, b, \xi) \in \mathbb{R}^{Q\times d} \times \mathbb{R}^Q \times \mathbb{R}_+^{n\times Q} : \\ \langle w_{y_i} - w_k, x_i\rangle + b_{y_i} - b_k \ge 1 - \xi_{ik}, \quad \forall 1 \le i \le n, 1 \le k \ne y_i \le Q \end{array} \right\},$$

and $\xi \in \mathbb{R}_+^{n\times Q}$ is a slack variable. In the objective function, $C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik}$ is the hinge loss term which presents the training classification errors. The regularizing term is known as a regularization. C is a parameter that presents the trade-off between the hinge loss and the regularizer term.

For feature selection in MSVM, we consider the two sparse MSVM models obtained from (2) by replacing the second term in the objective function with the ℓ_0 and/or the ℓ_2–ℓ_0 regularization, that lead to the so called ℓ_0-MSVM and ℓ_2–ℓ_0 -MSVM problems defined respectively by

$$\min_{(w,b,\xi)\in\Omega} C\sum_{i=1}^{n}\sum_{k\neq y_i} \xi_{ik} + \sum_{k=1}^{Q} \|w_k\|_0 \quad (\ell_0 - MSVM) \tag{3}$$

and

$$\min_{(w,b,\xi)\in\Omega} C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} + \beta\sum_{k=1}^{Q}\|w_k\|_2^2 + \sum_{k=1}^{Q}\|w_k\|_0 \quad (\ell_2-\ell_0-MSVM). \tag{4}$$

The backbone of our methods is DC programming and DCA whose brief overview will be given below.

2.2 A brief presentation of DC programming and DCA

DC programming and DCA constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. A general DC program takes the form:

$$\inf\{F(x) := G(x) - H(x) : x \in \mathbb{R}^n\}, \quad (P_{dc})$$

where G and H are lower semicontinuous proper convex functions on $\mathbb{R}^n$. Such a function F is called DC function, and $G - H$, DC decomposition of F while G and H are DC components of F. The convex constraint $x \in C$ can be incorporated in the objective function of (P_{dc}) by using the indicator function on C denoted χ_C which is defined by $\chi_C(x) = 0$ if $x \in C$; $+\infty$ otherwise:

$$\inf\{f(x) := G(x) - H(x) : x \in C\} = \inf\{\chi_C(x) + G(x) - H(x) : x \in \mathbb{R}^n\}.$$

A convex function θ is called *convex polyhedral* if it is the sum of the maximum of a finite set of affine functions and the indicator of a nonempty polyhedral convex set K, i.e.,

$$\theta(x) = \max_{i=1,\ldots,m}\{\langle a_i, x\rangle + b,\ a_i \in \mathbb{R}^n\} + \chi_K(x).$$

Polyhedral DC program occurs when either G or H is polyhedral convex. This class of DC programs, which is frequently encountered in practice, enjoys interesting properties (from both theoretical and practical viewpoints) concerning local optimality and the convergence of DCA (Le Thi and Pham Dinh 2005; Pham Dinh and Le Thi 1997).

A point x^* is said to be *a local minimizer* of $G - H$ if $G(x^*) - H(x^*)$ is finite and there exists a neighbourhood $\mathcal{U}$ of x^* such that

$$G(x^*) - H(x^*) \leq G(x) - H(x), \quad \forall x \in \mathcal{U}. \tag{5}$$

The necessary local optimality condition for (primal) DC program (P_{dc}) is given by

$$\emptyset \neq \partial H(x^*) \subset \partial G(x^*). \tag{6}$$

The condition (6) is also sufficient (for local optimality) in many important classes of DC programs, for example, when (P_{dc}) is a DC polyhedral program with H being polyhedral convex function, or when f is locally convex at x^* (see Le Thi and Pham Dinh 2005; Pham Dinh and Le Thi 1997, 1998).

A point x^* is said to be *a critical point* of $G - H$ if

$$\partial H(x^*) \cap \partial G(x^*) \neq \emptyset. \tag{7}$$

The relation (7) is in fact the generalized KKT condition for (P_{dc}) and x^* is also called a generalized KKT point.

DCA is based on local optimality conditions and duality in DC programming. The main idea of DCA is simple: each iteration of DCA approximates the concave part $-H$ by its affine majorization (that corresponds to taking $y^l \in \partial H(x^l)$) and minimizes the resulting convex function.

The generic DCA scheme can be described as follows:

DCA-General scheme
Initializations: let $x^0 \in \mathbb{R}^n$ be a best guess, $l \leftarrow 0$.
Repeat
1. Calculate $y^l \in \partial H(x^l)$.
2. Calculate $x^{l+1} \in arg\min\{G(x) - H(x^l) - \langle x - x^l, y^l\rangle : x \in \mathbb{R}^n\}$.
3. $l \leftarrow l + 1$.

Until convergence of $\{x^l\}$.

Convergence properties of DCA and its theoretical basics have been described in Le Thi and Pham Dinh (2005), Pham Dinh and Le Thi (1997, 1998). However, it is worthwhile to report the following properties that are useful in the next section (for simplicity's sake, we omit here the dual part of these properties).

i) DCA is a descent method (*without line search*): the sequences $\{G(x^l) - H(x^l)\}$ is decreasing.
ii) If $G(x^{l+1}) - H(x^{l+1}) = G(x^l) - H(x^l)$, then x^l is a critical point of $G - H$. In such a case, DCA terminates at l-th iteration.
iii) If the optimal value α of problem (P_{dc}) is finite and the sequences $\{x^l\}$ is bounded then every limit point x^* of the sequences $\{x^l\}$ is a critical point of $G - H$.
iv DCA has a *linear convergence* for general DC programs, and has a finite convergence for polyhedral DC programs.
v) If H is polyhedral convex and H is differentiable at x^*, then x^* is a local minimizer of (P_{dc}).

A deeper insight into DCA has been described in Le Thi and Pham Dinh (2005), Pham Dinh and Le Thi (1997), Pham Dinh and Le Thi (1998), Pham Dinh and Le Thi (2014). For instant it is crucial to note the main features of DCA: DCA is constructed from DC components and their conjugates but not the DC function f itself which has infinitely many DC decompositions, and there are as many DCA as there are DC decompositions. Such decompositions play a critical role in determining the speed of convergence, stability, robustness, and globality of sought solutions. It is important to study various equivalent DC forms of a DC program. This flexibility of DC programming and DCA is of particular interest from both a theoretical and an algorithmic point of view. Moreover, with suitable DC decompositions DCA generates most standard algorithms in convex and nonconvex optimization For a complete study of DC programming and DCA the reader is referred to Le Thi and Pham Dinh (2005), Pham Dinh and Le Thi (1997), Pham Dinh and Le Thi (1998), Pham Dinh and Le Thi (2014) and the references therein.

In the last decade, a variety of works in Machine Learning based on DC programming and DCA have been developed. The efficiency and the scalability of DCA have been proved in a lot of works (see e.g. Collobert et al. 2006; Krause and Singer 2004; Le Thi et al. 2006, 2007, 2008a, b, 2015a, b; Le Thi and Phan 2016a, b; Liu et al. 2005; Liu and Shen 2006; Neumann et al. 2005; Ong and Le Thi 2013; Le Thi and Pham Dinh 2005; Pham Dinh and Le Thi 1997, 1998, 2014; Ronan et al. 2006).

3 DC approximation approaches

For simplifying the presentation, we will consider the following common optimization problem:

$$\min \left\{ F(w,b,\xi) + \sum_{k=1}^{Q} \|w_k\|_0 : X = (w,b,\xi) \in \Omega \right\}, \tag{8}$$

where F stands for F_1 in the ℓ_0-MSVM problem, and for F_2 in the ℓ_2–ℓ_0-MSVM problem, say

$$F_1(w,b,\xi) := C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{ik}, \tag{9}$$

$$F_2(w,b,\xi) := C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{ik} + \beta \sum_{k=1}^{Q} \|w_k\|_2^2. \tag{10}$$

Here F_1 is a linear function while F_2 is a quadratic convex function.

We introduce in this section a class of DC approximation functions of the ℓ_0 norm. Define the step function $s : \mathbb{R} \to \mathbb{R}$ by

$$s(x) = 1 \text{ for } x \neq 0 \quad \text{and} \quad s(x) = 0 \text{ for } x = 0.$$

Then for $X \in \mathbb{R}^n$ we have $\|X\|_0 = \sum_{i=1}^{n} s(X_i)$. Let $\varphi_\theta : \mathbb{R} \to \mathbb{R}$ be a function depending on the parameter θ which approximates $s(x)$, say

$$\lim_{\theta \to +\infty} \varphi_\theta(x) = s(x), \ \forall x \in \mathbb{R}. \tag{11}$$

For symplifying the presentation, in the sequel, we will omit the parameter θ when this doesn't cause any ambiguity. Suppose that φ can be expressed as a DC function of the form

$$\varphi(x) = g(x) - h(x), \ x \in \mathbb{R} \tag{12}$$

where g and h are convex functions. Using this approximation, the ℓ_0 term in (8) can be written as

$$\sum_{k=1}^{Q} \|w_k\|_0 \approx \sum_{k=1}^{Q} \sum_{j=1}^{d} \varphi(w_{kj}) = \sum_{k=1}^{Q} \sum_{j=1}^{d} g(w_{kj}) - \sum_{k=1}^{Q} \sum_{j=1}^{d} h(w_{kj}), \tag{13}$$

and the problem (8) can be represented as follows:

$$\min \left\{ G(X) - H(X) : X \in \mathbb{R}^{Q \times d} \times \mathbb{R}^{Q} \times \mathbb{R}^{n \times Q} \right\}, \tag{14}$$

where

$$G(X) = \chi_\Omega(X) + F(X) + \sum_{k=1}^{Q} \sum_{j=1}^{d} g(w_{kj}); \quad H(X) = \sum_{k=1}^{Q} \sum_{j=1}^{d} h(w_{kj}).$$

Since the functions F, g and h are convex, G and H are convex too. Therefore (14) is a DC program. Thanks to the general DCA scheme given in Sect. 2, DCA applied on (14) can be described as follows.

DCA-dcApp
Initializations: Let τ be a tolerance sufficiently small, set $l = 0$. Let $X^0 = (w^0, b^0, \xi^0)$ be an initial point.
Repeat
Step 1. Compute $\overline{w}_{kj}^l \in \partial h(w_{kj}^l)\ \forall k = 1, \ldots, Q,\ j = 1, \ldots, d$ and set $Y^l = (\overline{w}^l, 0, 0)$.
Step 2. Compute $X^{l+1} = (w^{l+1}, b^{l+1}, \xi^{l+1})$ by solving the convex optimization problem

$$\min \left\{ F(X) + \sum_{k=1}^{Q}\sum_{j=1}^{d} g(w_{kj}) - \sum_{k=1}^{Q}\sum_{j=1}^{d} \overline{w}_{kj}^l w_{kj} : X = (w, b, \xi) \in \Omega \right\}. \tag{15}$$

Step 3. $l \leftarrow l + 1$.
Until $\| X^{l-1} - X^l \| \leq \tau(1 + \| X^l \|)$.

We consider now usual sparse inducing DC approximation functions φ and develop the corresponding **DCA-dcApp** to solve the resulting DC program.

First of all, we observe that the implementation of Algorithm **DCA-dcApp** according to each specific function φ differs from one to another by the computation of $\overline{w}_{kj}^l \in \partial h(w_{kj}^l)$ in the step 1, and the subproblem (15) in the step 2. On the other hand, with the same approximate function φ, **DCA-dcApp** applied on ℓ_0-MSVM and on ℓ_2–ℓ_0-MSVM problems share the same step 1 and distinguish only on the step 2. We will itemize below the computation of $\overline{w}_{kj}^l \in \partial h(w_{kj}^l)$ in the step 1 and the subproblem (15) in the step 2 of Algorithm **DCA-dcApp** in each specific case.

3.1 A piecewise exponential approximation

The piecewise exponential function introduced in Bradley and Mangasarian (1998) is defined as follows.

$$\varphi(x) = \begin{cases} 1 - \varepsilon^{-\alpha x} & \text{if } x \geq 0, \\ 1 - \varepsilon^{\alpha x} & \text{if } x < 0, \end{cases} \quad \alpha > 0.$$

φ can be expressed as a DC program of the form

$$\varphi(x) = g(x) - h(x), \quad g(x) = \max\{\alpha x, -\alpha x\}, \quad h(x) = \begin{cases} \alpha x - 1 + \varepsilon^{-\alpha x} & \text{if } x \geq 0 \\ -\alpha x - 1 + \varepsilon^{\alpha x} & \text{if } x \leq 0. \end{cases}$$

Clearly, the function h is differentiable and the computation of $\overline{w}_{kj}^l = \nabla h(w_{kj}^l)$ in the step 1 of **DCA-dcApp** is given by

$$\overline{w}_{kj}^l = \begin{cases} \alpha\left(1 - \varepsilon^{-\alpha w_{kj}^l}\right) & \text{if } w_{kj}^l \geq 0 \\ -\alpha\left(1 - \varepsilon^{\alpha w_{kj}^l}\right) & \text{if } w_{kj}^l < 0, \end{cases} \quad k = 1, \ldots, Q,\ j = 1, \ldots, d. \tag{16}$$

The step 2 of **DCA-dcApp** applied on the ℓ_0-MSVM (3) consists in solving the following convex program

$$\min_{(w,b,\xi)\in\Omega} C\sum_{i=1}^{n}\sum_{k\neq y_i} \xi_{ik} + \sum_{k=1}^{Q}\sum_{j=1}^{d} \max(\alpha w_{kj}, -\alpha w_{kj}) - \sum_{k=1}^{Q}\sum_{j=1}^{d} \overline{w}_{kj}^l w_{kj} \tag{17}$$

which can be transformed equivalently to the next linear program

$$\min_{w,b,\xi,t} \begin{cases} C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} + \sum_{k=1}^{Q}\sum_{j=1}^{d} t_{kj} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^{l} w_{kj} \\ \text{s.t. } (w,b,\xi)\in\Omega, \quad t\geq \alpha w,\ t\geq -\alpha w. \end{cases} \tag{18}$$

Finally, **DCA-dcApp** for solving the ℓ_0-MSVM problem (3) with piecewise exponential (PiE) approximation can be described as follows:

Algorithm 1: ℓ_0**-DCA_PiE**
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, \xi^0, b^0)$ be an initial point. Select α, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{w}^l$ via (16) and set $Y^l = (\overline{w}^l, 0, 0)$.
Step 2. Compute X^{l+1}, an optimal solution of the linear program (18).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \leq \epsilon\|X^l\|$. For ℓ_2–ℓ_0-MSVM problem (4), as indicated above, **DCA-dcApp** differs from Algorithm 1 by the convex subproblem in the step 2. It is now defined by

$$\min_{w,b,\xi,t} \begin{cases} \beta\sum_{k=1}^{Q}\|w_k\|_2^2 + C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} + \sum_{k=1}^{Q}\sum_{j=1}^{d} t_{kj} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^{l} w_{kj} \\ \text{s.t. } (w,b,\xi)\in\Omega,\ t\geq \alpha w, \quad t\geq -\alpha w. \end{cases} \tag{19}$$

Hence, **DCA-dcApp** for solving the ℓ_2–ℓ_0-MSVM problem (4) with PiE approximation is depicted below:

Algorithm 2: ℓ_2**–**ℓ_0**-DCA_PiE**
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, \xi^0, b^0)$ be an initial point. Select α, β, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{w}^l$ via (16) and set $Y^l = (\overline{w}^l, 0, 0)$.
Step 2. Compute X^{l+1}, an optimal solution of the convex quadratic program (19).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \leq \epsilon\|X^l\|$.

3.2 Capped-ℓ_1 approximation

The Capped-ℓ_1 approximation proposed in Peleg and Meir (2008) is given by

$$\varphi(x) = \min\{1, \alpha|x|\} = 1 + \alpha|x| - \max\{1, \alpha|x|\}, \quad \alpha > 0.$$

Let g and h be the functions defined by

$$g(x) = 1 + \alpha|x|, \quad h(x) = \max\{1, \alpha|x|\}.$$

Clearly, $\varphi = g - h$ and g, h are convex. Therefore φ is a DC function. The computation of $\overline{w}_{kj}^l \in \partial h(w_{kj}^l)$ in the step 1 of **DCA-dcApp** is given by

$$\overline{w}_{kj}^l = \begin{cases} 0 & \text{if } -1/\alpha \leq w_{kj}^l \leq 1/\alpha \\ \alpha & \text{if } w_{kj}^l > 1/\alpha \\ -\alpha & \text{if } w_{kj}^l < -1/\alpha \end{cases} \qquad k = 1,\ldots,Q,\ j = 1,\ldots,d. \tag{20}$$

Now, the convex subproblem in the step 2 of **DCA-dcApp** becomes

$$\min\left\{C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik}+\sum_{k=1}^{Q}\sum_{j=1}^{d}\alpha|w_{kj}|-\sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^{l}w_{kj}:(w,b,\xi)\in\Omega\right\}. \tag{21}$$

The last problem, like (17), is equivalent to the next linear program

$$\min_{w,b,\xi,t}\left\{\begin{array}{l} C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik}+\sum_{k=1}^{Q}\sum_{j=1}^{d}t_{kj}-\sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^{l}w_{kj}\\ \text{s.t. } (w,b,\xi)\in\Omega,\quad t\geq\alpha w, t\geq-\alpha w.\end{array}\right. \tag{22}$$

Hence, **DCA-dcApp** applied on (3) with Capped-ℓ_1 approximation is given below.

Algorithm 3: ℓ_0**-DCA_Cap-l1**
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, b^0, \xi^0)$ be an initial point. Select α, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{w}^l$ via (20) and set $Y^l = (\overline{w}^l, 0, 0)$.
Step 2. Compute X^{l+1}, an optimal solution of the linear program (22).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \leq \epsilon\|X^l\|$.

In case of ℓ_2–ℓ_0-MSVM problem (4), the convex subproblem takes the form

$$\min_{w,b,\xi,t}\left\{\begin{array}{l} C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik}+\beta\sum_{k=1}^{Q}\|w_k\|_2^2+\sum_{k=1}^{Q}\sum_{j=1}^{d}t_{kj}-\sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^{l}w_{kj}\\ \text{s.t. } (w,b,\xi)\in\Omega,\quad t\geq\alpha w, t\geq-\alpha w.\end{array}\right. \tag{23}$$

Then **DCA-dcApp** for solving (4) with Capped-ℓ_1 approximation can be presented as follows.

Algorithm 4: ℓ_2**–**ℓ_0**-DCA_Cap-l1**
Initializations: Let $\epsilon > 0$ be given and $X^0(w^0, b^0, \xi^0)$ be an initial point. Select α, β, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{w}^l$ via (20) and set $Y^l = (\overline{w}^l, 0, 0)$.
Step 2. Compute X^{l+1}, an optimal solution of the quadratic program (23).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \leq \epsilon\|X^l\|$.

3.3 A new piecewise linear approximation

We consider now a new and efficient approximation of ℓ_0-norm introduced in Le Thi (2012) (see also Le Thi et al. 2015b). Let a and b be positive constants, $0 \leq a < b$. Then the approximate function $\varphi(x)$ is defined as follows (Le Thi 2012):

$$\varphi(x) = \min\left\{1, \max\left\{0, \frac{|x|-a}{b-a}\right\}\right\}.$$

The function $\varphi(x)$ can be expressed as

$$\begin{aligned}\varphi(x) &= 1+\max\left(0, \tfrac{|x|-a}{b-a}\right)-\max\left(1, \tfrac{|x|-a}{b-a}\right)\\ &= \left(1+\tfrac{1}{b-a}\max(a,|x|)\right)-\left(\tfrac{1}{b-a}\max(b,|x|)\right).\end{aligned}$$

Let g and h be the functions defined by

$$g(x) = 1 + \frac{1}{b-a}\max(a, |x|); \quad h(x) = \frac{1}{b-a}\max(b, |x|).$$

They are clearly convex, and so, $\varphi(x) = g(x) - h(x)$ is a DC function.

Similarly to the case of Capped-ℓ_1 approximation, the computation of $\overline{w}_{kj}^l \in \partial h(w_{kj}^l)$ in the step 1 of **DCA-dcApp** is given by

$$\overline{w}_{kj}^l = \begin{cases} 0 & \text{if } |w_{kj}^l| \le b \\ \frac{1}{b-a} & \text{if } w_{kj}^l > b \\ \frac{-1}{b-a} & \text{if } w_{kj}^l < -b \end{cases} \qquad k = 1, \ldots, Q,\ j = 1, \ldots, d. \tag{24}$$

The convex subproblem at the step 2 of **DCA-dcApp** now has the form

$$\min_{(w,b,\xi)\in\Omega} \frac{1}{b-a}\sum_{k=1}^{Q}\sum_{j=1}^{d}\max(a, |w_{kj}|) + C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^l w_{kj}, \tag{25}$$

which is equivalent to the following linear program

$$\min_{w,b,\xi,t} \begin{cases} C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} + \frac{1}{b-a}\sum_{k=1}^{Q}\sum_{j=1}^{d} t_{kj} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^l w_{kj} \\ \text{s.t. } (w, b, \xi) \in \Omega, \quad t \ge a, t \ge w, t \ge -w. \end{cases} \tag{26}$$

The description of the **DCA-dcApp** for solving problem (3) with the piecewise linear (PiL) approximation is shown below.

Algorithm 5: ℓ_0**-DCA_PiL**
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, b^0, \xi^0)$ be an initial point. Select a, b, C and set $l = 0$;
Repeat
Step 1. Calculate $Y^l = (\overline{w}^l, 0, 0)$ via (24).
Step 2. Calculate X^{l+1}, an optimal solution of the linear program (26).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \le \epsilon\|X^l\|$.

For the ℓ_2–ℓ_0-MSVM problem (4), at the step 2 of **DCA-dcApp** we have to solve the following convex quadratic program

$$\min_{w,b,\xi,t} \begin{cases} \frac{1}{b-a}\sum_{k=1}^{Q}\sum_{j=1}^{d} t_{kj} + \beta\sum_{k=1}^{Q}\|w_k\|_2^2 + C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{w}_{kj}^l w_{kj} \\ \text{s.t. } (w, b, \xi) \in \Omega, \quad t \ge a,\ t \ge w,\ t \ge -w \end{cases}. \tag{27}$$

The description of **DCA-dcApp** for solving (4) with the PiL approximation is depicted in the algorithm 6 below.

Algorithm 6: ℓ_2**–**ℓ_0**-DCA_PiL**
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, b^0, \xi^0)$ be an initial point. Select a, b, β, C and set $l = 0$;
Repeat
1. Calculate $Y^l = (\overline{w}^l, 0, 0)$ via (24).
2. Calculate X^{l+1}, an optimal solution of the quadratic program (27).
3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \le \epsilon\|X^l\|$.

3.4 SCAD (Smoothly clipped absolute deviation) approximation

The well-known SCAD penalty function as studied in Fan and Li (2001) is defined by

$$\varphi(x) = \begin{cases} \lambda x & \text{if } 0 \le x \le \lambda, \\ -\frac{x^2 - 2\alpha\lambda x + \lambda^2}{2(\alpha-1)} & \text{if } \lambda < x \le \alpha\lambda, \\ \frac{(\alpha+1)\lambda^2}{2} & \text{if } x > \alpha\lambda, \\ \varphi(-x) & \text{if } x < 0, \end{cases}$$

where $\lambda > 0$ and $\alpha > 2$ are parameters. Let g and h be the functions given by

$$g(x) = \lambda|x| \quad \text{and} \quad h(x) = \begin{cases} 0 & \text{if } 0 \le x \le \lambda \\ \frac{1}{2(\alpha-1)}(x-\lambda)^2 & \text{if } \lambda < x \le \alpha\lambda \\ \lambda x - \frac{(\alpha+1)\lambda^2}{2} & \text{if } x > \alpha\lambda \\ h(-x) & \text{if } x < 0. \end{cases}$$

It is easy to verify that g and h are convex functions and $\varphi(x) = g(x) - h(x)$, so φ is a DC function. Moreover, h is differentiable and the computation of $\overline{w}_{kj}^l = \nabla h(w_{kj}^l)$ in the step 1 of **DCA-dcApp** is given by the following expression

$$\overline{w}_{kj}^l = \begin{cases} 0 & \text{if } |w_{kj}^l| \le \lambda \\ (w_{kj}^l - \lambda)/(\alpha-1) & \text{if } \lambda < w_{kj}^l \le \alpha\lambda \\ (w_{kj}^l + \lambda)/(\alpha-1) & \text{if } -\alpha\lambda \le w_{kj}^l < -\lambda \\ \lambda & \text{if } w_{kj}^l > \alpha\lambda \\ -\lambda & \text{if } w_{kj}^l < -\alpha\lambda \end{cases} \quad k = 1, \ldots, Q;\ j = 1, \ldots, d. \tag{28}$$

As with the cases of PiL approximation and Capped-ℓ_1 approximation, the step 2 of **DCA-dcApp** applied to problem (3) consists of solving the following linear program

$$\min_{w,b,\xi,t} \left\{ \begin{array}{l} C \sum_{i=1}^n \sum_{k \ne y_i} \xi_{ik} + \sum_{k=1}^Q \sum_{j=1}^d t_{kj} - \sum_{k=1}^Q \sum_{j=1}^d \overline{w}_{kj}^l w_{kj} \\ \text{s.t. } (w, b, \xi) \in \Omega, \quad t \ge \lambda w, t \ge -\lambda w. \end{array} \right. \tag{29}$$

Finally, the **DCA-dcApp** applied to the ℓ_0-MSVM problem (3) with the SCAD approximation is described as follows:

Algorithm 7: ℓ_0-DCA_SCAD
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, b^0, \xi^0)$ be an initial point. Select α, λ, C and set $l = 0$;
Repeat
Step 1. Calculate $Y^l = (\overline{w}^l, 0, 0)$ via (28).
Step 2. Calculate X^{l+1}, an optimal solution of the linear program (29).
Step 3. $l \leftarrow l + 1$.
Until $\|X^{l-1} - X^l\| \le \epsilon \|X^l\|$.

For problem (4), we need only to replace the linear program in the step 2 of Algorithm 7 with the following convex quadratic program

$$\min_{w,b,\xi,t} \left\{ \begin{array}{l} C \sum_{i=1}^n \sum_{k \ne y_i} \xi_{ik} + \beta \sum_{k=1}^Q \|w_k\|_2^2 + \sum_{k=1}^Q \sum_{j=1}^d t_{kj} - \sum_{k=1}^Q \sum_{j=1}^d \overline{w}_{kj}^l w_{kj} \\ \text{s.t. } (w, b, \xi) \in \Omega, \quad t \ge \lambda w, t \ge -\lambda w. \end{array} \right. \tag{30}$$

Hence, the **DCA-dcApp** for solving the ℓ_2–ℓ_0-MSVM problem (4) with SCAD approximation is presented in the algorithm 8 below.

Algorithm 8: ℓ_2–ℓ_0-DCA_SCAD
Initializations: Let $\epsilon > 0$ be given and $X^0 = (w^0, b^0, \xi^0)$ be an initial point. Select $\alpha, \lambda, \beta, C$ and set $l = 0$;
Repeat
1. Calculate $Y^l = (\overline{w}^l, 0, 0)$ via (28).
2. Calculate X^{l+1}, an optimal solution of the convex quadratic program (30).
3. $l \leftarrow l + 1$.

Until $\|X^{l-1} - X^l\| \leq \epsilon \|X^l\|$.

3.5 A logarithm approximation

Consider now the logarithm (Log) approximation function defined by

$$\varphi(x) = \rho_\tau \log\left(1 + \frac{|x|}{\tau}\right),$$

where $\rho_\tau = \frac{1}{\log\left(1+\frac{1}{\tau}\right)}$ with $\tau > 0$ being a parameter. This approximation has been used for feature selection in Weston et al. (2003) and compressed sensing in Candès et al. (2008).

Since $\log(\cdot)$ is an increasing function, by introducing the variable $v \in \mathbb{R}_+^{Q\times d}$ we can rewrite the problem (8) as

$$\min_{(w,b,\xi,v)\in\widetilde{\Omega}} F(w, b, \xi) + \sum_{k=1}^{Q}\sum_{j=1}^{d} \varphi(v_{kj}), \tag{31}$$

where $\widetilde{\Omega} = \{(w, b, \xi, v) : (w, b, \xi) \in \Omega, -v \leq w \leq v\}$.

Let $g(x) = 0$ and $h(x) = -\rho_\varepsilon \log(1 + x/\varepsilon)$. Clearly, $\varphi(x) = g(x) - h(x)$ and g, h are convex functions on $\mathbb{R}_+$. Then we can express the problem (31) as a DC program

$$\min\left\{\widetilde{G}(\widetilde{X}) - \widetilde{H}(\widetilde{X}) : \widetilde{X} = (w, b, \xi, v) \in \mathbb{R}^{Q\times d+Q+n\times Q+Q\times d}\right\}, \tag{32}$$

where $\widetilde{G}(\widetilde{X}) = \chi_{\widetilde{\Omega}}(\widetilde{X}) + F(w, b, \xi)$ and $\widetilde{H}(\widetilde{X}) = \sum_{k=1}^{Q}\sum_{j=1}^{d} h(v_{kj})$ are convex functions on $\widetilde{\Omega}$.

DCA applied on the problem (31) amounts to computing, at each iteration l, a subgradient $\widetilde{Y}^l = (0, 0, 0, \overline{v}^l) \in \partial\widetilde{H}(\widetilde{X}^l)$ with $\overline{v}_{kj}^l = \partial h(v_{kj}^l)$ $(k = 1, \ldots, Q,\ j = 1, \ldots, d)$ and then, solving the convex program

$$\min\{F(w, b, \xi) - \langle\widetilde{Y}^l, \widetilde{X}\rangle : \widetilde{X} = (w, b, \xi, v) \in \widetilde{\Omega}\}. \tag{33}$$

It can be seen that $\widetilde{H}$ is differentiable and $\nabla\widetilde{H}(w, b, \xi, v^l) = (0, 0, 0, \overline{v}^l)$, where

$$\overline{v}_{kj}^l = \nabla h\left(v_{kj}^l\right) = -\frac{\rho_\varepsilon}{v_{kj}^l + \tau}, \quad k = 1, \ldots, Q,\ j = 1, \ldots, d. \tag{34}$$

For the ℓ_0-MSVM problem (3), i.e. F stands for F_1 in (31), problem (33) becomes the following linear program

$$\min\left\{C\sum_{i=1}^{n}\sum_{k\neq y_i}\xi_{ik} - \sum_{k=1}^{Q}\sum_{j=1}^{d}\overline{v}_{kj}^l v_{kj} : (w, b, \xi, v) \in \widetilde{\Omega}\right\}. \tag{35}$$

Hence, DCA for solving the ℓ_0-MSVM problem (3) with Log approximation can be described as follows.

Algorithm 9: ℓ_0-DCA_Log
Initialization Let ϵ be a sufficiently small tolerance, and $\widetilde{X}^0 = (w^0, b^0, \xi^0, v^0)$ be a guess. Select τ, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{v}^l$ via (34).
Step 2. Compute $\widetilde{X}^{l+1} = \left(w^{k+1}, b^{k+1}, \xi^{k+1}, v^{k+1}\right)$ by solving linear program (35).
Step 3. $l \leftarrow l + 1$.
Until $\|\widetilde{X}^{l-1} - \widetilde{X}^l\| \leq \epsilon \|\widetilde{X}^l\|$.

For the ℓ_2–ℓ_0-MSVM problem (4), F will stand for F_2 in (31). Thus, problem (33) now becomes the following quadratic program

$$\min \left\{ C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{ik} + \beta \sum_{k=1}^{Q} \|w_k\|_2^2 - \sum_{k=1}^{Q} \sum_{j=1}^{d} \overline{v}_{kj}^l v_{kj} : (w, b, \xi, v) \in \widetilde{\Omega} \right\}. \tag{36}$$

Consequently, DCA for solving the ℓ_2–ℓ_0-MSVM problem (4) with Log approximation is presented below.

Algorithm 10: ℓ_2–ℓ_0-DCA_Log
Initializations: Let $\epsilon > 0$ be given and $\widetilde{X}^0 = \left(w^0, b^0, \xi^0, v^0\right)$ be an initial point. Select τ, β, C and set $l = 0$;
Repeat
Step 1. Compute $\overline{v}^l$ via (34).
Step 2. Compute $\widetilde{X}^{l+1} = \left(w^{k+1}, b^{k+1}, \xi^{k+1}, v^{k+1}\right)$, an optimal solution of the convex quadratic program (36).
Step 3. $l \leftarrow l + 1$.
Until $\|\widetilde{X}^{l-1} - \widetilde{X}^l\| \leq \epsilon \|\widetilde{X}^l\|$.

3.6 Convergence properties

Theorem 1 (Convergence properties of the above 10 DCA based algorithms)

(i) The sequence $\{G(X^l) - H(X^l)\}$ *(resp.* $\{\widetilde{G}(\widetilde{X}^l) - \widetilde{H}(\widetilde{X}^l)\}$*) is monotonously decreasing.*

(ii) In algorithms ℓ_0***-DCA_PiE***, ℓ_0***-DCA_Cap-l1***, ℓ_2–ℓ_0***-DCA_Cap-l1***, ℓ_0***-DCA_PiL***, ℓ_2–ℓ_0***-DCA_PiL***, ℓ_0***-DCA_SCAD*** *(resp. algorithm* ℓ_0***-DCA_Log****), the sequences* $\{X^l = (w^l, \xi^l, b^l)\}$ *and* $\{Y^l = (\overline{w}^l, 0, 0)\}$ *(resp.* $\{\widetilde{X}^l = (w^l, b^l, \xi^l, v^l)\}$ *and* $\{\widetilde{Y}^l = (0, 0, 0, \overline{v}^l)\}$*) converge respectively to* $X^* = (w^*, \xi^*, b^*)$ *and* $Y^* = (\overline{w}^*, 0, 0)$ *(resp.* $\widetilde{X}^* = (w^*, b^*, \xi^*, v^*)$ *and* $\widetilde{Y}^* = (0, 0, 0, \overline{v}^*)$*) after a finite number of iterations. Moreover,* X^* *(resp.* $\widetilde{X}^*$*) is almost always a local minimizer of problem (3) (resp. (31)).*
Especially, for algorithms ℓ_0***-DCA_Cap-l1*** *and* ℓ_2–ℓ_0***-DCA_Cap-l1*** *(resp.* ℓ_0***-DCA_PiL*** *and* ℓ_2–ℓ_0***-DCA_PiL****), if*

$$w_{kj}^* \notin \left\{-\frac{1}{\alpha}, \frac{1}{\alpha}\right\} \left(\text{resp. } w_{kj}^* \notin \{-b, b\}\right) \quad \forall k = 1, \ldots, Q,\ j = 1, \ldots, d, \tag{37}$$

then X^* *is actually a local minimizer of problem (3).*

Proof (i) is a consequence of the DCA's convergence property i) mentioned in Sect. 2. We are going to prove (ii).

For algorithm ℓ_0-**DCA_Cap-l1**, the second DC component of (14), says H, is polyhedral convex, so (14) is a polyhedral DC program. By using the DCA's convergence properties iv) and v) mentioned in Sect. 2, ℓ_0-**DCA_Cap-l1** has finite convergence; moreover, if H is differentiable at X^*, e.g. if the condition (37) holds, then X^* is a local minimizer of problem (3). Since a polyhedral convex function is almost always differentiable, say, it is differentiable everywhere except on a set of measure zero, we can say that X^* is almost always a local minimizer of problem (3).

The same arguments are also applied to the cases of ℓ_2–ℓ_0-**DCA_Cap-l1**, ℓ_0-**DCA_PiL** and ℓ_2–ℓ_0-**DCA_PiL**.

For algorithm ℓ_0-**DCA_PiE**, since the first DC component G is polyhedral convex, (14) is a DC polyhedral program, so ℓ_0-**DCA_PiE** has a finite convergence. We consider the dual problem of (14) in which the second DC component G^*, the conjugate[1] of G, is polyhedral convex. By the same arguments as for algorithm ℓ_0-**DCA_Cap-l1**, we conclude that Y^* is almost always a local minimizer of the dual problem. According to the property of transportation of local minimizers in DC programming (see Pham Dinh and Le Thi 1997; Le Thi and Pham Dinh 2005), we deduce that X^* is almost always a local minimizer of problem (3).

For algorithms ℓ_0-**DCA_SCAD** and ℓ_0-**DCA_Log**, the proof is similar. □

4 A continuous reformulation approach via exact penalty techniques

In this section we reformulate equivalently the problem (8) in the form of a DC program and develop two DCA based algorithms for solving it. Denote by e the vector of ones in the appropriate space.

Let $u \in \mathbb{R}^{Q\times d}$ be the binary variable defined by:

$$u_{kj} = \begin{cases} 1 & \text{if } w_{kj} \neq 0 \\ 0 & \text{if } w_{kj} = 0, \end{cases} \quad \forall k = 1, \ldots;\ Q,\ j = 1, \ldots, d. \tag{38}$$

We have

$$\sum_{k=1}^{Q} \|w_k\|_0 = \sum_{k=1}^{Q}\sum_{j=1}^{d} u_{kj} = \langle e, u\rangle. \tag{39}$$

Suppose that Ω is bounded in the variable w, i.e. $\Omega \subset [-\mathcal{B}, \mathcal{B}]^{Q\times d} \times \mathbb{R}^Q \times \mathbb{R}_+^{n\times Q}$ for some $\mathcal{B} > 0$. Then the problem (8) can be expressed as

$$\begin{cases} \min_{(w,b,\xi,u)} & F(w, b, \xi) + \langle e, u\rangle \\ \text{s.t.} & (w, b, \xi) \in \Omega, \\ & |\ w_{kj}\ | \leq \mathcal{B}u_{kj},\ u_{kj} \in \{0, 1\}, \quad \forall k = 1, \ldots, Q;\ j = 1, \ldots, d. \end{cases} \tag{40}$$

Let $\overline{p} : \mathbb{R}^{Q\times d} \times \mathbb{R}^Q \times \mathbb{R}^{n\times Q} \times [0, 1]^{Q\times d} \to \mathbb{R}$ be the function defined as $\overline{p}(w, b, \xi, u) = p(u)$ with $p : \mathbb{R}^{Q\times d} \to \mathbb{R},\ p(u) := \sum_{k=1}^{Q}\sum_{j=1}^{d} u_{kj}(1 - u_{kj})$, and let Λ be the polyhedral convex set determined by

$$\Lambda := \left\{(w, b, \xi, u) \in \Omega \times [0, 1]^{Q\times d} : |w_{kj}| \leq |\mathcal{B}u_{kj}|,\ \forall k = 1, \ldots, Q;\ j = 1, \ldots, d, \right\}. \tag{41}$$

[1] The conjugate G^* of a convex function G is defined by $G^*(Y) := \sup_X \{\langle X, Y\rangle - G(X)\}$.

We observe that $\overline{p}$ is concave and $\overline{p}(w, b, \xi, u) \geq 0 \forall (w, b, \xi, u) \in \Lambda$. By dint of the exact penalty technique developed recently in Le Thi et al. (2012), the problem (40) is equivalent to the following continuous optimization problem, with sufficient large positive numbers $\eta > \eta_0 \geq 0$ (called penalty parameters)

$$\min \{F(w, b, \xi) + \langle e, u\rangle + \eta p(u) : \mathcal{X} = (w, b, \xi, u) \in \Lambda\} \tag{42}$$

We investigate now a DCA scheme for solving (42). Let $\overline{G}$ and $\overline{H}$ be the functions defined by

$$\overline{G}(\mathcal{X}) := F(X) + \chi_\Lambda(\mathcal{X})$$

and

$$\overline{H}(\mathcal{X}) := \eta \sum_{k=1}^{Q}\sum_{j=1}^{d} u_{kj}^2 - (\eta + 1)\sum_{k=1}^{Q}\sum_{j=1}^{d} u_{kj}.$$

The problem (42) can be expressed as:

$$\min \left\{\overline{G}(\mathcal{X}) - \overline{H}(\mathcal{X}) : \mathcal{X} = (w, b, \xi, u) \in \mathbb{R}^{Q\times d+Q+n\times Q+Q\times d}\right\}. \tag{43}$$

Obviously, $\overline{G}$ and $\overline{H}$ are convex functions and so (43) is a DC program. DCA applied on (43) consists of computing, at each iteration l,

$$\mathcal{Y}^l \in \partial\overline{H}(\mathcal{X}^l),\ \mathcal{X}^{l+1} \in \arg\min \left\{\overline{G}(\mathcal{X}) - \langle \mathcal{Y}^l, \mathcal{X}\rangle : \mathcal{X} \in \Lambda\right\}.$$

Clearly, $\overline{H}$ is differentiable and $\mathcal{Y}^l = \nabla\overline{H}(\mathcal{X}^l)$ can be computed directly

$$\mathcal{Y}^l = (0, 0, 0, \overline{u}^l),\ \text{with } \overline{u}_{kj}^l = 2\eta u_{kj}^l - (\eta + 1), \quad \forall k = 1, \ldots, Q;\ j = 1, \ldots, d. \tag{44}$$

And $\mathcal{X}^{l+1} = (w^{l+1}, b^{l+1}, \xi^{l+1}, u^{l+1})$ is an optimal solution of the convex optimization problem

$$\min \left\{F(X) - \langle \overline{u}^l, u\rangle : \mathcal{X} \in \Lambda\right\}. \tag{45}$$

Hence, the DCA applied to (43) when $F = F_1$ (the ℓ_0 -MSVM problem) is described below.

ℓ_0-DCA-Econ
Initializations: Let $\tau > 0$ be given and $\mathcal{X}^0 = (w^0, b^0, \xi^0, u^0)$ be an initial point. Select $\eta, \mathcal{B}, C$ and set $l = 0$.
Repeat
Step 1. Calculate $\mathcal{Y}^l$ via (44).
Step 2. Calculate $\mathcal{X}^{l+1}$, an optimal solution of the linear program

$$\min \left\{ C\sum_{i=1}^{n}\sum_{k\neq y_i} \xi_{ik} - \langle \overline{u}^l, u\rangle : \mathcal{X} = (w, b, \xi, u) \in \Lambda \right\}.$$

Step 3. $l \leftarrow l + 1$.
Until $\|\mathcal{X}^{l-1} - \mathcal{X}^l\| \leq \tau\|\mathcal{X}^l\|$.

Remark 1 Note that since $\overline{G}$ is polyhedral convex, (43) is a polyhedral DC program and then **ℓ_0-DCA-Econ** has a finite convergence. Furthermore, by considering the dual problem of (43) in which the second DC component is polyhedral convex and using the property v) mentioned in Sect. 2, we can prove that **ℓ_0-DCA-Econ** converges almost always to a local minimizer of (43).

Similarly, the DCA applied to (43) when $F = F_2$ (the ℓ_2–ℓ_0-MSVM problem) is described as follows:

ℓ_2–ℓ_0-DCA-Econ
Initializations: Let $\tau > 0$ be given and $\mathcal{X}^0 = (w^0, b^0, \xi^0, u^0)$ be an initial point. Select $\eta, \mathcal{B}, \beta, C$ and set $l = 0$.
Repeat
Step 1. Calculate $\mathcal{Y}^l$ via (44).
Step 2. Calculate $\mathcal{X}^{l+1}$, an optimal solution of the convex quadratic program

$$\min \left\{ C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{ik} + \beta \sum_{k=1}^{Q} \|w_k\|_2^2 - \langle \overline{u}^l, u \rangle : \mathcal{X} = (w, b, \xi, u) \in \Lambda \right\}.$$

Step 3. $l \leftarrow l + 1$.
Until $\|\mathcal{X}^{l-1} - \mathcal{X}^l\| \leq \tau \|\mathcal{X}^l\|$.

5 Numerical experiments

We have implemented the algorithms in the V.S C++ v6.0 environment and performed the experiments a Intel Core^{TM} I7 (2×2.2 Ghz) processor, 4 GB RAM. The purpose is to clarify the multiple questions: the ℓ_0 or ℓ_2–ℓ_0 regularization which one is better? What is the approach advised—the DC approximation or the continuous exact reformulation? And in the DC approximation approaches, what is the best approximation among several sparse inducing functions? Is there the consistency among features that are selected from different DCA based algorithms?

5.1 Datasets

We consider eight popular datasets often used for feature selection: Lung Cancer (LUN), Optical Recognition of Handwritten Digits (OPT), Libras Movement (MOV), Semeion Handwritten Digit (SEM), Multiple Features (MFE), CNAE-9 (CNA), Internet Advertisement (ADV), and ADN. The first 7 datasets are taken from UCI Machine Learning Repository while the last can be found at ftp://genbank.bio.net. Each dataset is divided into two parts—the training set and the test set, except for the LUN dataset which contains a very small number of samples, therefore the whole dataset is used as the training set as well as the test set. For the OPT dataset, both training set and test set are given in UCI Machine Learning Repository. For the other 6 datasets, training and test sets are randomly sampled from the original data with 60 % for training and the remaining 40 % for testing. We repeat 10 times this random procedure and get 10 couple of training/test sets of each dataset. These datasets are described in details in the Table 1.

5.2 Experiment setting

The CPLEX 12.5 solver is used to solve linear and convex quadratic problems.

The parameters are taken as follows: for all methods, the most appropriate values of the parameter C are chosen by a five-folds cross-validation; β, the coefficient of the ℓ_2 term is set to 0.01; the tolerance τ (in the stopping criterion of DCA) is set to 10^{-6}.

Observe that, theoretically speaking, the larger value of α is, the better DC approximation of ℓ_0-norm would be, while practically, when α is large, the algorithms give sometimes bad

Table 1 The description of the datasets

Dataset	#Feature	#Class	#Train	#Test
LUN	56	3	16	16
OPT	63	10	3823	1797
ADN	60	3	1913	1273
MOV	90	15	225	135
SEM	256	10	960	633
MFE	649	10	1200	800
CNA	856	9	648	432
ADV	1558	2	1967	1312

local minima. Hence, we use an α updating procedure during the algorithms. Starting with a small value of α_0, we increase it at each iteration l by $\alpha_{l+1} = \alpha_l + \Delta\alpha$ until a given threshold $\overline{\alpha}$. For ℓ_0-DCA_PiE and ℓ_2–ℓ_0-DCA_PiE, $\alpha_0 = 1.5$ and $\Delta\alpha = 0.5$, $\overline{\alpha} = 5.5$. For ℓ_0-DCA_Cap-l1 and ℓ_2–ℓ_0-DCA_Cap-l1 we set $\alpha_0 \in \{0.7, 0.8, 0.9\}$, $\Delta\alpha = 0.2$, $\overline{\alpha} = 5.5$. For ℓ_0-DCA_PiL and ℓ_2–ℓ_0-DCA_PiL, we take $a = 10^{-6}$ and $b \in \{10^{-4}, 10^{-3}, \ldots, 10^{-1}, 0.2, 0.3\}$. The parameters α and λ in ℓ_0-DCA_SCAD and ℓ_2–ℓ_0-DCA_SCAD are, respectively, set to 3.4 and 0.4 as proposed by Fan and Li (2001). In ℓ_0-DCA_Log and ℓ_2–ℓ_0-DCA_Log, ε is set to 10^{-4}. Finally, for ℓ_0-DCA_Econ and ℓ_2 -ℓ_0-DCA_Econ, the starting value of η is chosen in $\{10, 20, \ldots, 50\}$ and η is doubled at each iteration until 10, 000. The parameter $\mathcal{B}$ is set to 1000.

The starting point w^0, u^0 and v^0 of the DCA based algorithms are randomly chosen in $[-0.5, 0.5]^{Q\times d}$.

We compare our methods with one of the best algorithms for feature selection in MSVM, called the Adaptive Sub-Norm (ASN) method (see Zhang et al. 2008 for more details). To select relevant features, we first compute the feature ranking score c_j, $j = 1, \ldots, d$ for each feature (Duan et al. 2005) as follows

$$c_j = \sum_{i=1}^{Q} |w_{ij}|.$$

This ranking score is then normalized. Let $\varsigma = \max_j c_j$, we compute $c_j = \frac{c_j}{\varsigma}$ for each $j = 1, \ldots, d$. Then, we remove the features j for which c_j is smaller than a given threshold (0.01 in our experiments). After removing features, for computing the accuracy of classifier, we apply again ℓ_2-MSVM (2) on the new training datasets and calculate the classification's accuracy on the new test sets.

5.3 Numerical results

The comparative results of 12 versions of the DCA based algorithms and the concurrent ASN method are presented in Table 2 (the number and the percentage of selected features) and Table 3 (the accuracy of classifiers and the CPU time in seconds). On each given couple of training/test set, each algorithm is performed 10 times. In these tables, the average result (on 10 running times for LUN and OPT, and on 100 running times for the six remaining data) and its standard deviation are reported.

For an easy observation, the number of selected features and the corresponding accuracy of classifiers on the ℓ_0 model are shown in the Figs. 1 and 2.

Table 2 Number and % of selected features of 12 versions of DCA applied on two regularization models (ℓ_0 and ℓ_2–ℓ_0) with 5 approximations (PiE, Cap-l1, PiL, SCAD, Log) and the exact penalty continuous approach (Econ), and the Adaptive Sub-norm (ASN)—one of the best related algorithms

Data	Model	PiE	Cap-l1	PiL	SCAD	Log	Econ	ASN
Number of selected features								
LUN	ℓ_0	**4.38** ± 0.48	5.02 ± 1.11	7.3 ± 1.27	7.02 ± 0.88	7.84 ± 0.79	8.00 ± 0.84	11.2 ± 1.17
	$\ell_2\ell_0$	**4.53** ± 0.50	6.40 ± 1.20	7.69 ± 0.94	7.00 ± 1.19	5.00 ± 0.30	9.02 ± 1.16	
OPT	ℓ_0	25.82 ± 2.36	31.78 ± 1.21	33.07 ± 0.80	26.78 ± 0.63	**22.58** ± 0.91	26.96 ± 0.56	53 ± 0.89
	$\ell_2\ell_0$	25.73 ± 1.24	32.31 ± 1.98	34.31 ± 0.66	46.82 ± 0.57	**22.47** ± 1.02	27.87 ± 1.20	
ADN	ℓ_0	7.10 ± 2.71	**6.98** ± 0.58	12.04 ± 0.92	14.18 ± 0.82	12.00 ± 0.73	13.07 ± 0.85	18.30 ± 0.64
	$\ell_2\ell_0$	14.72 ± 1.77	**6.19** ± 1.20	18.02 ± 0.41	14.98 ± 0.41	12.13 ± 0.63	13.02 ± 0.33	
MOV	ℓ_0	24.27 ± 1.85	23.53 ± 0.98	33.13 ± 0.69	26.69 ± 0.84	31.24 ± 1.32	**23.27** ± 1.36	37.90 ± 0.94
	$\ell_2\ell_0$	32.07 ± 4.80	25.62 ± 0.48	34.34 ± 0.80	32.78 ± 1.49	31.04 ± 0.42	**24.98** ± 0.54	
SEM	ℓ_0	73.04 ± 1.11	**37.16** ± 0.76	79.87 ± 1.15	67.73 ± 0.49	83.42 ± 1.14	70.40 ± 1.27	202.4 ± 3.83
	$\ell_2\ell_0$	74.53 ± 1.86	**38.2** ± 0.54	81.00 ± 1.46	70.38 ± 1.08	82.96 ± 0.89	82.78 ± 0.87	
MFE	ℓ_0	49.78 ± 2.32	47.84 ± 1.13	**15.91** ± 0.81	49.27 ± 1.00	34.76 ± 0.97	51.20 ± 1.26	59 ± 0.77
	$\ell_2\ell_0$	51.09 ± 1.75	48.18 ± 0.85	51.42 ± 1.24	49.44 ± 1.34	**35.02** ± 1.02	54.47 ± 0.83	
CNA	ℓ_0	58.80 ± 5.87	78.98 ± 0.91	52.09 ± 1.36	77.80 ± 1.13	**16.04** ± 0.76	45.18 ± 1.40	102.9 ± 1.51
	$\ell_2\ell_0$	59.08 ± 4.26	79.67 ± 1.25	64.09 ± 0.69	77.91 ± 0.69	**27.04** ± 1.38	44.93 ± 1.06	
ADV	ℓ_0	11.02 ± 1.61	30.56 ± 1.22	30.96 ± 1.03	20.93 ± 0.57	6.02 ± 0.71	**5.18** ± 0.71	52.8 ± 1.89
	$\ell_2\ell_0$	11.46 ± 2.20	30.80 ± 1.22	14.00 ± 0.73	31.11 ± 1.49	6.04 ± 0.63	**5.91** ± 0.63	
Selected features (%)								
LUN	ℓ_0	**7.82** ± 0.86	8.96 ± 1.98	13.04 ± 2.27	12.54 ± 1.57	14.00 ± 1.41	14.29 ± 1.50	20.00 ± 2.09
	$\ell_2\ell_0$	**8.09** ± 0.89	11.43 ± 2.14	13.73 ± 1.68	12.50 ± 2.13	8.93 ± 0.54	16.11 ± 2.07	
OPT	ℓ_0	40.98 ± 3.75	50.44 ± 1.92	52.49 ± 1.27	42.51 ± 1.00	**35.84** ± 1.44	42.79 ± 0.89	84.13 ± 1.41
	$\ell_2\ell_0$	40.84 ± 1.97	51.29 ± 3.14	54.46 ± 1.05	74.32 ± 0.90	**35.67** ± 1.62	44.24 ± 1.90	

Table 2 continued

Data	Model	PiE	Cap-l1	PiL	SCAD	Log	Econ	ASN
ADN	ℓ_0	11.83 ± 4.52	**11.63** ± 0.97	20.07 ± 1.53	23.63 ± 1.37	20.00 ± 1.22	21.78 ± 1.42	30.50 ± 1.07
	$\ell_2\ell_0$	24.53 ± 2.95	**10.32** ± 2.00	30.03 ± 0.68	24.97 ± 0.68	20.22 ± 1.05	21.70 ± 0.55	
MOV	ℓ_0	26.97 ± 2.06	26.14 ± 1.09	36.81 ± 0.77	29.66 ± 0.93	34.71 ± 1.47	**25.86** ± 1.51	42.11 ± 1.04
	$\ell_2\ell_0$	35.63 ± 5.33	28.47 ± 0.53	38.16 ± 0.89	36.42 ± 1.66	34.49 ± 0.47	**27.76** ± 0.60	
SEM	ℓ_0	28.53 ± 0.43	**14.52** ± 0.30	31.20 ± 0.45	26.46 ± 0.19	32.59 ± 0.45	27.50 ± 0.50	79.06 ± 1.50
	$\ell_2\ell_0$	29.11 ± 0.73	**14.92** ± 0.21	31.64 ± 0.57	27.49 ± 0.42	32.41 ± 0.35	32.34 ± 0.34	
MFE	ℓ_0	7.67 ± 0.36	7.37 ± 0.17	**2.45** ± 0.12	7.59 ± 0.15	5.36 ± 0.15	7.89 ± 0.19	9.09 ± 0.12
	$\ell_2\ell_0$	7.87 ± 0.27	7.42 ± 0.13	7.92 ± 0.19	7.62 ± 0.21	**5.40** ± 0.16	8.39 ± 0.13	
CNA	ℓ_0	6.87 ± 0.69	9.23 ± 0.11	6.09 ± 0.16	9.09 ± 0.13	**1.87** ± 0.09	5.28 ± 0.16	12.02 ± 0.18
	$\ell_2\ell_0$	6.90 ± 0.50	9.31 ± 0.15	7.49 ± 0.08	9.10 ± 0.08	**3.16** ± 0.16	5.25 ± 0.12	
ADV	ℓ_0	0.71 ± 0.10	1.96 ± 0.08	1.99 ± 0.07	1.34 ± 0.04	0.39 ± 0.05	**0.33** ± 0.05	3.39 ± 0.12
	$\ell_2\ell_0$	0.74 ± 0.14	1.98 ± 0.08	0.90 ± 0.05	2.00 ± 0.10	0.39 ± 0.04	**0.38** ± 0.04	

The bold fonts correspond to the best results

Table 3 Classification accuracy and CPU time of 12 versions of DCA applied on two regularization models (ℓ_0 and ℓ_2–ℓ_0) with 5 approximations (PiE, Cap-l1, PiL, SCAD, Log) and the exact penalty continuous approach (Econ), and the Adaptive Sub-norm (ASN)—one of the best related algorithms

Data	Model	PiE	Cap-l1	PiL	SCAD	Log	Econ	ASN
Accuracy of classifiers (%)								
LUN	ℓ_0	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00
	$\ell_2\ell_0$	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	68.75 ± 0.00	
OPT	ℓ_0	93.55 ± 0.17	93.95 ± 1.01	**95.25** ± 0.56	94.58 ± 0.15	92.28 ± 0.51	94.12 ± 0.85	93.23 ± 0.46
	$\ell_2\ell_0$	93.59 ± 0.16	93.88 ± 0.69	**95.81** ± 0.11	94.35 ± 0.15	92.28 ± 0.19	94.87 ± 1.01	
ADN	ℓ_0	86.86 ± 0.97	83.21 ± 0.98	85.88 ± 2.23	**87.07** ± 0.64	86.75 ± 1.12	86.12 ± 0.98	86.51 ± 0.67
	$\ell_2\ell_0$	**87.26** ± 0.88	83.19 ± 1.51	85.81 ± 2.56	87.21 ± 0.73	85.45 ± 3.18	86.61 ± 3.35	
MOV	ℓ_0	**75.93** ± 3.95	69.29 ± 0.36	75.68 ± 1.02	73.49 ± 0.53	70.12 ± 0.98	72.22 ± 0.54	71.02 ± 0.89
	$\ell_2\ell_0$	72.31 ± 0.98	69.84 ± 1.21	**76.58** ± 0.89	73.53 ± 1.54	69.98 ± 1.33	73.44 ± 1.64	
SEM	ℓ_0	81.89 ± 0.16	**90.55** ± 0.26	83.01 ± 0.25	79.89 ± 0.56	82.56 ± 0.54	82.55 ± 0.32	86.14 ± 0.66
	$\ell_2\ell_0$	82.11 ± 0.21	**90.57** ± 0.56	83.15 ± 0.11	82.51 ± 0.63	84.56 ± 0.25	82.88 ± 0.52	
MFE	ℓ_0	95.73 ± 0.15	96.15 ± 0.42	96.35 ± 0.55	96.22 ± 0.27	95.33 ± 0.15	**96.88** ± 0.52	95.68 ± 0.32
	$\ell_2\ell_0$	95.88 ± 0.22	**96.89** ± 0.18	96.88 ± 0.19	96.44 ± 0.29	95.31 ± 0.15	96.35 ± 0.29	
CNA	ℓ_0	86.99 ± 1.66	90.59 ± 0.56	89.33 ± 0.96	**90.68** ± 0.51	76.57 ± 0.18	86.67 ± 0.95	90.12 ± 0.74
	$\ell_2\ell_0$	85.44 ± 2.13	**90.31** ± 0.52	89.55 ± 0.54	90.15 ± 0.89	84.47 ± 1.59	86.33 ± 0.68	
ADV	ℓ_0	**95.47** ± 1.01	93.64 ± 0.68	93.25 ± 0.45	94.28 ± 0.77	94.52 ± 1.15	93.66 ± 0.51	93.42 ± 0.54
	$\ell_2\ell_0$	**95.94** ± 0.58	93.66 ± 0.51	95.12 ± 0.69	95.14 ± 0.69	94.33 ± 0.69	93.98 ± 0.58	
CPU time (s)								
LUN	ℓ_0	0.11 ± 0.02	0.07 ± 0.03	0.11 ± 0.04	0.13 ± 0.02	**0.05** ± 0.06	0.09 ± 0.03	0.11 ± 0.02
	$\ell_2\ell_0$	0.12 ± 0.03	0.12 ± 0.05	0.14 ± 0.02	0.13 ± 0.01	**0.02** ± 0.01	0.07 ± 0.02	
OPT	ℓ_0	89.88 ± 1.46	**55.05** ± 8.12	186.57 ± 3.52	56.88 ± 2.23	199.9 ± 1.02	104.35 ± 1.65	425.98 ± 0.77
	$\ell_2\ell_0$	156.51 ± 3.13	185.68 ± 0.89	185.65 ± 4.21	**39.98** ± 1.59	429.35 ± 8.89	205.32 ± 1.52	

Table 3 continued

Data	Model	PiE	Cap-l1	PiL	SCAD	Log	Econ	ASN
ADN	ℓ_0	**1.77** $\pm$ 0.62	3.39 $\pm$ 0.38	2.85 $\pm$ 0.15	3.32 $\pm$ 0.27	4.89 $\pm$ 1.33	2.32 $\pm$ 0.56	11.21 $\pm$ 0.69
	$\ell_2\ell_0$	42.53 $\pm$ 4.18	95.35 $\pm$ 10.26	30.25 $\pm$ 1.15	21.32 $\pm$ 1.98	19.66 $\pm$ 0.25	**4.99** $\pm$ 10.23	
MOV	ℓ_0	48.52 $\pm$ 10.82	160.35 $\pm$ 3.35	**0.59** $\pm$ 0.24	45.36 $\pm$ 11.25	0.61 $\pm$ 0.19	6.52 $\pm$ 0.98	25.56 $\pm$ 0.71
	$\ell_2\ell_0$	48.18 $\pm$ 14.81	166.53 $\pm$ 5.69	72.58 $\pm$ 1.11	52.39 $\pm$ 6.85	**0.89** $\pm$ 0.15	35.92 $\pm$ 22.89	
SEM	ℓ_0	145.86 $\pm$ 1.88	77.64 $\pm$ 1.13	172.56 $\pm$ 4.12	155.85 $\pm$ 2.86	25.35 $\pm$ 1.22	**12.35** $\pm$ 0.77	169.99 $\pm$ 0.56
	$\ell_2\ell_0$	351.25 $\pm$ 3.32	189.65 $\pm$ 3.25	309.25 $\pm$ 1.99	351.35 $\pm$ 1.89	103.73 $\pm$ 2.35	**44.39** $\pm$ 1.82	
MFE	ℓ_0	300.5 $\pm$ 18.6	**70.68** $\pm$ 1.56	421.26 $\pm$ 2.89	302.56 $\pm$ 7.54	93.26 $\pm$ 0.31	432.6 $\pm$ 12.25	441.32 $\pm$ 1.01
	$\ell_2\ell_0$	531.21 $\pm$ 7.22	152.96 $\pm$ 1.26	655.25 $\pm$ 3.59	512.39 $\pm$ 7.62	201.36 $\pm$ 1.91	**125.39** $\pm$ 1.57	
CNA	ℓ_0	11.85 $\pm$ 4.81	60.51 $\pm$ 5.64	11.52 $\pm$ 0.36	11.21 $\pm$ 0.67	**3.12** $\pm$ 0.11	12.23 $\pm$ 1.54	42.97 $\pm$ 0.96
	$\ell_2\ell_0$	55.17 $\pm$ 9.76	61.25 $\pm$ 5.78	**15.68** $\pm$ 0.52	53.22 $\pm$ 3.12	20.55 $\pm$ 0.32	18.56 $\pm$ 8.95	
ADV	ℓ_0	4.11 $\pm$ 0.18	3.55 $\pm$ 0.58	3.21 $\pm$ 0.32	3.35 $\pm$ 0.15	**1.81** $\pm$ 0.24	2.78 $\pm$ 0.77	13.38 $\pm$ 0.64
	$\ell_2\ell_0$	4.88 $\pm$ 0.79	5.83 $\pm$ 1.89	3.01 $\pm$ 0.46	5.12 $\pm$ 0.56	**1.36** $\pm$ 0.09	4.45 $\pm$ 1.52	

The bold fonts correspond to the best results

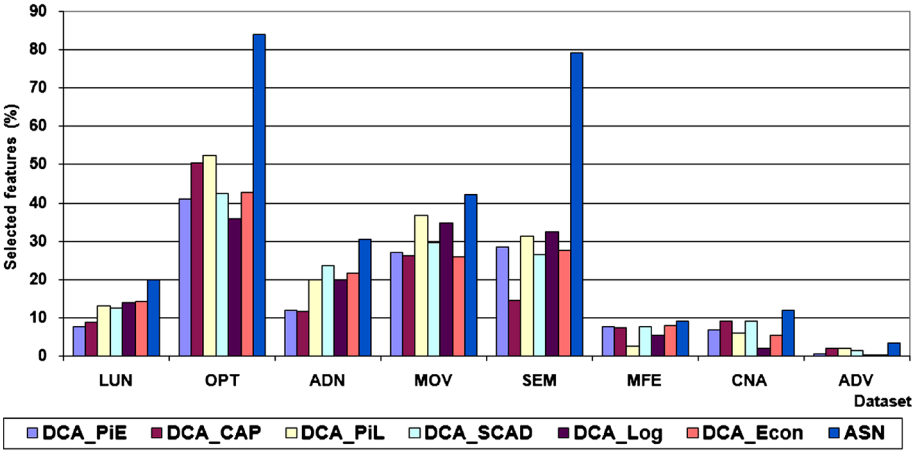

Fig. 1 Selected features (%) of ℓ_0-DCAs and ASN

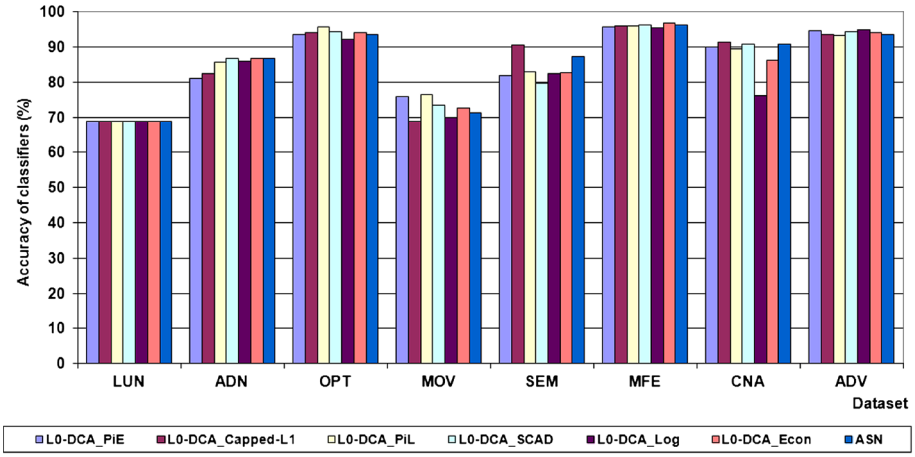

Fig. 2 Accuracy of classifiers (%) of ℓ_0-DCAs and ASN

It is interesting to study the consistency of features that are selected from different DCA based algorithms. In the six last columns of Table 4 we report, respectively, the number of features selected by six (6Algo), five (5Algo), four (4Algo), three (3Algo), two (2Algo), one (1Algo) algorithm(s). Here "Best" (resp. "Worst") stands for the number of selected features of the best (resp. worst) algorithm in terms of sparsity.

5.4 Comments on numerical results

Generally speaking, all the DCA based algorithms applied on the ℓ_0 models (ℓ_0-DCA) give a good sparsity of solutions. The percentage of selected features varies from 0.33 to 52.49 %. For the ℓ_2–ℓ_0 models, the DCA based algorithms (ℓ_2–ℓ_0-DCA) give similar results to (but slightly less good than) those of ℓ_0-DCA. Both ℓ_0-DCA and ℓ_2–ℓ_0-DCA not only provide a good performance in term of feature selection, but also give a high accuracy of classifiers (from 68.75 to 96.88 % for the ℓ_0 model and from 68.75 to 96.89 % for the ℓ_2–ℓ_0 model).

Table 4 Frequency of selected features by 6 DCA based algorithms

Dataset	Best	Worst	6Algo	5Algo	4Algo	3Algo	2Algo	1Algo
LUN	4	9	2	2	3	2	3	1
OPT	20	37	19	5	4	5	6	5
ADN	5	15	5	1	1	5	0	4
MOV	23	33	4	9	10	8	9	15
SEM	21	70	10	4	4	16	14	31
MFE	15	53	11	19	3	18	23	0
CNA	15	76	4	27	4	7	17	26
ADV	5	33	2	4	5	9	12	21

More specifically, it is worth to mention the following observations which contribute to clarify the several questions indicated at the beginning of this section.

5.4.1 DCA based algorithms versus ASN

All the DCA based algorithms are better than ASN in terms of feature selection. The gain of sparsity (the difference of the percentage of selected features) of ℓ_0-DCA (resp. ℓ_2–ℓ_0-DCA) varies from 1.2 to 46.47 % (resp. from 0.47 to 46.45 %). The accuracy of classifier of DCA based algorithms is also better than ASN in most of cases (except for the logarithm approximation Log), and the best DCA version is always better than ASN on all datasets. More precisely, ℓ_0-DCA (resp. ℓ_2–ℓ_0-DCA) with PiE, Cap-l1, PiL, SCAD approximation and the exact penalty continuous approach (Econ) is better than ASN on 6/8, 6/8, 4/8, 7/8, and 5/8 (resp. 6/8, 6/8, 5/8, 7/8, and 6/8) datasets, respectively. As for the computation time, all ℓ_0-DCAs are generally faster than ASN. More precisely, ℓ_0-DCA_Log and ℓ_0-DCA_Econ are faster than ASN on 8/8 datasets and the 4 remaining ℓ_0-DCAs are faster than ASN on 6/8 datasets. The gain ratio grows up to 43.32 times (ℓ_0-DCA_PiL on MOV dataset). Likewise, all (resp. 5/6) ℓ_2–ℓ_0-DCAs are faster than ASN on ADV (resp. OPT) dataset. On the remaining datasets, some ℓ_2–ℓ_0-DCAs are more expensive than ASN: 3/6 (resp. 4/6) ℓ_2–ℓ_0-DCAs are slower than ASN on CNA and MFE (resp. MOV and SEM) datasets. The size of the LUN dataset is very small, hence the difference on CPU running time between the algorithms is not significant.

5.4.2 Approximation approaches versus the exact penalty approach

Numerically, the results are comparable in terms of sparsity and classification accuracy, i.e. there is no a great difference between these two approaches. This can be justified by a theoretical result proved in Le Thi et al. (2015b): with a suitable parameter, the resulting approximation problems using Cap-l1 or SCAD are equivalent to the exact penalized continuous problem (see Le Thi et al. 2015b for more details). Meanwhile, in some cases the exact penalty approach is very fast, the gain ratio can grow up to 12 times (SEM dataset, in comparing with ℓ_0-DCA_PiE).

5.4.3 ℓ_0 regularization versus ℓ_2–ℓ_0 regularization

Based on the same approximation function, ℓ_0-DCAs are better than ℓ_2–ℓ_0-DCAs in term of sparsity on 7/8 (resp. 4/8) datasets with PiE, Cap-l1, PiL, SCAD (resp. Log). Likewise,

ℓ_0-DCA_Econ is better than ℓ_2–ℓ_0-DCA_Econ on 6/8 datasets. Meanwhile, the accuracy of classifiers of ℓ_2–ℓ_0-DCAs are slightly better than that of ℓ_0-DCAs (in 6/8 datasets with PiE, SCAD, Econ, 5/8 datasets with Capped-ℓ_1, PiL, and 4/8 datasets with Log). This result justifies the utility of the ℓ_2–ℓ_0 regularization, in particular to overcome overfitting. As for the rapidity, not surprisingly, the ℓ_0-DCAs are faster than ℓ_2–ℓ_0-DCAs, because that ℓ_0-DCAs solve one linear program while ℓ_2–ℓ_0-DCAs solve one convex quadratic program at each iteration.

5.4.4 About sparse inducing functions

The classification accuracy of DCA based algorithms with different approximation functions are comparable: the difference from one to another is less than 3 %, except for SEM dataset where Cap-l1 is considerably better than the others (from 6.45 to 10.66 %). Cap-l1, followed by PiL, gives the best accuracy in most of cases.

However the approximation functions influenced more considerably on the sparsity: the difference of selected feature varies from 1.16 to 18 %.

Overall, Cap-l1 seems to be the best approximation in the sense that it realizes a good trade-off between sparsity and accuracy. Once again, this result confirms the consistency property of Cap-l1 proved in Le Thi et al. (2015b).

Concerning the computation time, the comparative results are quite different among the height datasets and there is no "winner" on all datasets. Each of ℓ_0-DCA_Log and ℓ_2–ℓ_0-DCA_Log is the fastest on 3 datasets. For the remaining dataset, each of DCA-PiE, DCA-Cap-l1, DCA-PiL and DCA-SCAD is the fastest one time.

5.4.5 The consistency of the selected features from different versions of DCA

The results in Table 4 show that there is a consistency of the selected features from different versions of DCA. Some features are selected by all or most of all algorithms. The high frequency selection of the features allows us to identify "important" features.

5.4.6 The connection between the testing datasets and different versions of DCA

We identify only one "strong" connection: ℓ_0-DCA_Cap-l1 and ℓ_2–ℓ_0-DCA_Cap-l1 are preferred to SEM dataset. Indeed, DCA_Cap-l1 are considerably better than the others on both classification accuracy (from 6.47 to 10.66 %) and sparsity (from 11.94 to 18.04 %). Otherwise, we can say that ℓ_0-DCA_PiL is preferred to MFE dataset since it is better than the others, from 2.91 to 5.44 % on sparsity and in terms of classification accuracy ℓ_0-DCA_PiL and the exact penalty approach give the best results (with a litle difference 0.53 %).

As for the relation between the dataset and CPU running time, for the multi-classification (i.e. the number of classes is at least 3), the CPU running time depends on the size (the number of classes $\times$ the number of features $\times$ the number of samples in the train set) of the dataset. In general, with the same model (ℓ_0 or ℓ_2–ℓ_0 regularization) the larger size of dataset is, the more important computation time will be. Indeed, we observe that for all algorithms (except for ℓ_0-DCA-Cap-l1) the MFE dataset ($10 \times 649 \times 1200$) is the most expensive among 8 datasets, followed by SEM ($10 \times 256 \times 960$) and then OPT ($10 \times 63 \times 3823$). The CNA dataset ($9 \times 856 \times 464$) is special, its size is quite large but is not expensive.

Finally, we observe that DCA based algorithms work well on both balanced (the sizes of classes are relatively equal) and unbalanced datasets. For instance, these algorithms give

good results for the ADV dataset which has the size of two classes are respectively 319 and 1960.

6 Conclusion

We have developed efficient approaches based on DC programming and DCA for feature selection in multi-class support vector machine. Based on appropriate approximation functions of zero-norm and an exact penalty technique, the ℓ_0-MSVM and ℓ_2–ℓ_0-MSVM problems are reformulated as DC programs. It fortunately turns out that the corresponding DCA consists in solving, at each iteration, one linear program (in ℓ_0 regularization) or one convex quadratic program (in ℓ_2–ℓ_0 regularization). Moreover, several DCA based algorithms converge, after a finite number of iterations, almost always to a local solution. Numerical results on several real datasets showed the robustness, the effectiveness of the DCA based schemes. We are convinced that DCA is a promising approach for feature selection in MSVM.

By proposing twelve DCA based algorithms we offered to the users a large choice of suitable algorithms according to their learning situations. From both theoretical and algorithmic points of view, ℓ_0-DCA is a very efficient approach because that it results in polyhedral DC programs for which the corresponding DCA requires solving one linear program at each iteration, and converges after a finite number of iterations to not only a critical point but also a local minimizer in almost always cases. Especially, the sparse inducing functions such as Capped-ℓ_1 and Piecewise linear are interesting since an explicit local sufficient condition is available. Overall, to get a good trade-off between sparsity and accuracy by a fast algorithm, the ℓ_0-DCA-Cap-l1 as well as the exact penalty approach ℓ_0-DCA-Econ are the best choices. In particular, Capped-ℓ_1 followed by Piecewise linear approximation are very recommended while Logarithm function is to avoid when the classification accuracy is the most important criterion, and ℓ_2–ℓ_0-DCA is not recommended either when the sparsity is the most important criterion or for massive datasets.

Another important contribution of this paper lies on the determination of the most important features in the dataset via the high frequency of selected features when performing several DCA based algorithms.

This research can be extended to the nonlinear separable case by introducing a kernel in the MSVM model. Work in this direction is in progress.

Acknowledgements This research is funded by Foundation for Science and Technology Development of Ton Duc Thang University (FOSTECT), website: http://fostect.tdt.edu.vn, under Grant FOSTECT.2015.BR.15. The authors would like to thank the referees and the guest editor for their valuable comments which helped to improve the manuscript.

References

Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In J. Shavlik (Ed.), *Machine learning proceedings of the fifteenth international conferences (ICML'98)* (pp. 82–90). San Francisco: Morgan Kaufmann.

Cai, X., Nie, F., Huang, H., & Ding, C. (2011). Multi-class $\ell_{2,1}$-norm support vector machine. In *Data mining (ICDM), 2011 IEEE 11th International Conference* (pp. 91–100).

Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, *14*, 877–905.

Chapelle, O. (2008). *Multi-class feature selection with support vector machines*. Technical report YR-2008-002.

Chen, Y. W., & Lin, C. J. (2006). Combining SVMs with various feature selection strategies. In I. Guyon, M. Nikravesh, S. Gunn, & L. A. Zadeh (Eds.), *Feature extraction. Studies in Fuzziness and Soft Computing* (Vol. 207, pp. 315–324). Berlin: Springer.

Chen, Y., Li, Y., Cheng, X-Q., & Guo, L. (2006). Survey and taxonomy of feature selection algorithms in intrusion detection system. In *Proceedings of Inscrypt 2006, LNCS 4318* (pp. 153–167).

Chen, X., Zeng, X., & Alphen, D. V. (2006). Multi-class feature selection for texture classification. *Pattern Recognition Letters*, *27*, 1685–1691.

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive SVMs. *Journal of Machine Learning Research*, *7*, 1687–1712.

Deng, S., Xu, Y., Li, L., Li, X., & He, Y. (2013). A feature-selection algorithm based on Support Vector Machine-multiclass for hyperspectral visible spectral analysis. *Journal of Food Engineering*, *119*(1), 159–166.

Duan, K. B., Rajapakse, J. C., Wang, H., & Azuaje, F. (2005). Multiple SVM-RFE for genne selection in cancer classification with expression data. *IEEE Transactions on Nanobioscience*, *4*, 228–234.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, *96*, 1348–1360.

Gribonval, R., & Nielsen, M. (2003). Sparse representation in union of bases. *IEEE Transactions on Information Theory*, *49*, 3320–73325.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (2006). *Feature extraction, foundations and applications*. Berlin: Springer.

Hermes, L., & Buhmann, J. M. (2000). Feature selection for support vector machines. *Proceedings of the 15th International Conference on Pattern Recognition, vol. 2* (pp. 712–715).

Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, *13*(2), 415–425.

Huang, J., Ma, S., & Zhang, C. H. (2008). Adaptive Lasso for sparse high-dimentional regression models. *Statistica Sinica*, *18*, 1603–1618.

Huang, L., Zhang, H. H., Zeng, Z. B., & Bushel, P. R. (2013). Improved sparse multi-class SVM and its application for gene selection in cancer classification. *Cancer Inform*, *12*, 143–153.

Hui, Z. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, *101*(476), 1418–1429.

Krause, N., & Singer, Y. (2004). Leveraging the margin more carefully. In *Proceeding of ICML '04* (pp. 63–71). NY, USA.

Le Thi, H. A. (2005). DC programming and DCA. Available on http://lita.sciences.univ-metz.fr/~lethi/DCA.html.

Le Thi, H. A. (2012). *A new approximation for the ℓ_0-norm*. Research Report LITA EA 3097, University of Lorraine.

Le Thi, H. A., & Phan, D. N. (2016). DC programming and DCA for sparse fisher linear discriminant analysis. *Neural Computing and Applications*, doi:10.1007/s00521-016-2216-9.

Le Thi, H. A., Belghiti, T., & Pham Dinh, T. (2006). A new efficient algorithm based on DC programming and DCA for Clustering. *Journal of Global Optimization*, *37*, 593–608.

Le Thi, H. A., Le Hoai, M., & Dinh, T. Pham. (2015). Feature Selection in machine learning: An exact penalty approach using a Difference of Convex function algorithm. *Machine Learning*, *101*(1–3), 163–186.

Le Thi, H. A., Le Hoai, M., Nguyen, V. V., & Pham Dinh, T. (2008). A DC programming approach for feature selection in Support Vector Machines learning. *Journal of Advances in Data Analysis and Classification*, *2*(3), 259–278.

Le Thi, H. A., Le Hoai, M., & Pham Dinh, T. (2007). Optimization based DC programming and DCA for hierarchical clustering. *European Journal of Operational Research*, *183*, 1067–1085.

Le Thi, H. A., Huynh, V. N., & Pham Dinh, T. (2012). Exact penalty and error bounds in DC programming. *Journal of Global Optimization*, *52*(3), 509–535.

Le Thi, H. A., Nguyen, V. V., & Ouchani, S. (2008). Gene selection for cancer classification using DCA. In C. Tang, C. X. Ling, X. Zhou, N. J. Cercone, & X. Li (Eds.), *ADMA 2008. LNCS (LNAI)* (Vol. 5139, pp. 62–72). Heidelberg: Springer.

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

Le Thi, H. A., Pham Dinh, T., Le Hoai, M., & Vo, X. T. (2015). DC approximation approaches for sparse optimization. *European Journal of Operational Research*, *244*(1), 26–46.

Le Thi, H. A., & Phan, D. N. (2016). DC programming and DCA for sparse optimal scoring problem. *Neurocomputing*, *186*, 170–181.

Lee, Y., Kim, Y., Lee, S., & Koo, J. (2006). Structured multicategory support vector machines with analysis of variance decomposition. *Biometrika*, *93*(3), 555–71.

Lee, Y., Lin, Y., & Wahba, G. (2004). Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, *99*(465), 67–81.

Li, G. Z., Yang, J., Liu, G. P., & Xue, L. (2004). Feature selection for multi-class problems using support vector machines. In *PRICAI 2004: Trends in artificial intelligence, lecture notes in computer science 3157* (pp. 292–300). Berlin: Springer.

Liu, D., Qian, H., Dai, G., & Zhang, Z. (2013). An iterative SVM approach to feature selection and classification in high-dimensional datasets. *Pattern Recognition*, *46*(9), 2531–2537.

Liu, Y., & Shen, X. (2006). Multicategory Ψ-learning. *Journal of the American Statistical Association*, *101*(474), 500–509.

Liu, Y., Shen, X., & Doss, H. (2005). Multicategory ψ-learning and Support Vector Machine: Computational tools. *Journal of Computational and Graphical Statistics*, *14*, 219–236.

Liu, Y., Zhang, H. H., Park, C., & Ahn, J. (2007). Support vector machines with adaptive ℓ_q penalty. *Computational Statistics & Data Analysis*, *51*, 6380–6394.

Maldonado, S., Weber, R., & Basak, J. (2011). Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences*, *181*(1), 115–128.

Neumann, J., Schnörr, C., & Steidl, G. (2005). Combined SVM-based feature selection and classification. *Machine Learning*, *61*(1–3), 129–150.

Ong, C. S., & Le Thi, H. A. (2013). Learning sparse classifiers with Difference of Convex functions algorithms. *Optimization Methods and Software*, *28*, 4.

Peleg, D., & Meir, R. (2008). A bilinear formulation for vector sparsity optimization. *Signal Processing*, *8*(2), 375–389.

Pham Dinh, T., & Le Thi, H. A. (2014). Recent advances on DC programming and DCA. In *Transactions on computational intelligence XIII, Lecture Notes in Computer Science Vol. 8342* (pp. 1–37).

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

Pham Dinh, T., & Le Thi, H. A. (1998). Optimization algorithms for solving the trust region subproblem. *SIAMJ. Optimization*, *2*, 476–505.

Rakotomamonjy, A. (2003). Variable selection using SVM-based criteria. *Journal of Machine Learning Research*, *3*, 1357–1370.

Ramona, M., Richard, G., & David, B. (2012). Multiclass feature selection with kernel gram-matrix-based criteria. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(10), 1611–1623.

Ronan, C., Fabian, S., Jason, W., & Lé, B. (2006). Trading convexity for scalability. In *Proceedings of the 23rd international conference on machine learning ICML 2006* (pp. 201–208). Pittsburgh, Pennsylvania.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, *46*, 431–439.

Wang, H., Li, G., & Jiang, G. (2007). Robust regression shrinkage and consistent variable selection via the LAD-LASSO. *Journal of Business & Economics Statistics*, *25*(3), 347–355.

Wang, L., & Shen, X. (2003). On ℓ_1-norm multi-class support vector machine: Methodology and theory. *Journal of the American Statistical Association*, *102*, 583–594.

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings-European symposium on artificial neural networks, ESANN 1999* (pp. 219–224). D-Facto public.

Weston, J., Elisseeff, A., & Schölkopf, B. (2003). Use of zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461.

Wu, K., Lu, B., Uchiyama, M. & Isahara, H. (2007). A probabilistic approach to feature selection for multi-class text categorization. In D. Liu et al. (Eds.), *ISNN 2007, Part I, LNCS 4491* (pp. 1310–1317).

Yeh, Y., Chung, Y., Lin, T., & Wang, Y. (2011). Group lasso regularized multiple kernel learning for heterogeneous feature selection. In *The 2011 international joint conference on neural networks (IJCNN)* (pp. 2570–2577).

Zhang, H. H., Liu, Y., Wu, Y., & Zhu, J. (2008). Variable selection for the multicategory SVM via adaptive sup-norm regularization. *Journal of Statistics*, *2*, 149–167.

Zhou, Y., Jin, R. & Hoi, S. C. (2010). Exclusive lasso for multi-task feature selection. In *AISTATS 9*.

Zhou, X., & Tuck, D. P. (2007). MSVM-RFE: Extensions of SVM-RFE for multiclass gene selection on DNA microarray data. *Bioinformatics*, *23*(9), 1106–1114.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, *101*, 1418–71429.