*Optimization Methods and Software*
Vol. 00, No. 00, January 2011, 1–26

# *Learning sparse classifiers with Difference of Convex functions Algorithms*

Cheng Soon Ong[a] , Le Thi Hoai An[b*]

[a]*Department of Computer Science, ETH Zurich, Switzerland*;
[b]*Laboratory of Theoretical and Applied Computer Science, University of Paul Verlaine - Metz, Ile de Saulcy, 57045 Metz, France*;
(*Received 00 Month 200x; in final form 00 Month 200x*)

Sparsity of a classifier is a desirable condition for high dimensional data and large sample sizes. This paper investigates the two complementary notions of sparsity for binary classification: sparsity in the number of features and sparsity in the number of examples. Several different losses and regularizers are considered: the hinge loss and ramp loss, and $\ell_2$, $\ell_1$, approximate $\ell_0$, and capped $\ell_1$ regularization. We propose three new objective functions that further promote sparsity, the capped $\ell_1$ regularization with hinge loss, and the ramp loss versions of approximate $\ell_0$ and capped $\ell_1$ regularization. We derive difference of convex functions algorithms (DCA) for solving these novel non-convex objective functions. The proposed algorithms are shown to converge in a finite number of iterations to a local minimum. Using simulated data and several datasets from the UCI machine learning repository, we empirically investigate the fraction of features and examples required by the different classifiers.

**Keywords:** Sparse features and examples, binary classification, DCA.

**AMS Subject Classification**:

## 1. Introduction and background

We study the effect of different losses and regularization terms on binary classification. The task consists of discriminating between two classes of examples, by convention called "positive" (+1) and "negative" (-1), based on some observed features. The learning algorithm is given a training set consisting of $n$ example-label pairs $(x_i, y_i)$, and the aim is to estimate the parameters $w, b$ of the model such that a certain objective function is minimized. In this paper, we consider linear classifiers $\langle w, x\rangle + b$ with an objective function of the form

$$\min_{w,b} \sum_{i=1}^{n} \ell(y_i, x_i; w, b) + \lambda\Omega(w), \tag{1}$$

where $\ell$ is a loss function, $\lambda$ is a regularization parameter, and $\Omega(\cdot)$ is the regularizer (or penalty term). This general form of an objective function has been traditionally used with convex loss functions and convex regularizers. For example, the popular support vector machine (SVM) uses the hinge loss and the squared $\ell_2$ norm regularizer (see Section 1.1). Further background can be found in [1, 2] and references therein.

This paper was presented the International Conference on Optimization: Techniques and Applications (ICOTA 8) in Shanghai, 10-13 December 2010.
*Corresponding author. Email: lethi@univ-metz.fr

ISSN: 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/1055678xxxxxxxxxxxxx
http://www.informaworld.com

Table 1. Summary of algorithms considered. CapOneNormSVM, RLSVM0 and RLSVMC1 are new objective functions proposed in this paper.

| | Hinge Loss | Ramp Loss |
|---|---|---|
| $\ell_2$ | SVM [2] | RLSVM2 [5] |
| $\ell_1$ | OneNormSVM [15] | RLSVM1 [7] |
| $\ell_0$ | ZeroNormSVM [10] | **RLSVM0** (Section 2.4.1) |
| Capped $\ell_1$ | **CapOneNormSVM** (Section 2.4.2) | **RLSVMC1** (Section 2.4.3) |

Recently, there has been renewed interest in non-convex losses. For example in boosting, it has been shown that convex potential functions are not robust against label noise [3] and a non-convex potential can mitigate this problem [4]. For support vector machines (SVM), recent work has increased sparsity by using the truncated loss [5, 6]. In [7], it is shown that the truncation results in Fisher consistency of multiclass classifiers. In addition, the truncated hinge loss results in tighter learning theoretic bounds [8, 9]. Parallel to this, there has been recent work on non-convex regularizers to better approximate the $\ell_0$ norm regularization [10–12].

However the resulting optimization problems are non-convex and in general difficult to solve. In this paper, we develop an unified approach based on difference of convex (DC) programming and DC algorithms (DCA). DCA is a fast and scalable approach for non-convex and non-smooth optimization [13, 14]. Generally, DCA aims to solve a DC program that takes the form:

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

where $G, H$ are lower semicontinuous proper convex functions on $\mathbb{R}^m$. Such functions $F$ are called DC functions, and $G - H$ is called the DC decomposition of $F$ while $G$ and $H$ are DC components of $F$. Hence, for a DC program, each DC decomposition corresponds to a different version of DCA. A DC function $F$ has an infinite number of DC decompositions which influence the quality of the resulting algorithm, for example speed of convergence, robustness, efficiency and globality of computed solutions. Therefore, the search for a "good" DC decomposition is important from algorithmic point of view, and is dependent on the specific structure of the problem being considered.

We consider eight objective functions in this paper, which corresponds to: setting the loss to the hinge loss and the ramp loss; and setting the regularizer to $\ell_2$ and $\ell_1$ norms, and $\ell_0$ and capped $\ell_1$ functions. The objective functions are summarised in Table 1, and are explained in detail in Section 2. The upper two objective functions in the left column (SVM and OneNormSVM) are convex and the rest are non-convex.

The contributions of this paper are as follows: We provide a unified analysis of the non-convex objectives for binary classification via DCA. This results in a novel efficient algorithm for CapOneNormSVM. We propose three new sparse objectives (CapOneNormSVM, RLSVM0 and RLSVMC1), and derive efficient DCAs to solve them. In a sense, we are filling an obvious gap in the class of problems of the form of Equation (1). The three proposed DCA schemes have interesting convergence properties: the algorithms converge to a critical point after finitely many iterations, and it is almost always case this point is a local minimizer to the considered problem. The details of the algorithms and their convergence are discussed in Section 3.2. Moreover, the algorithms consist of solving one linear program at each iteration. We perform a careful empirical comparison of the different objectives (Section 4), and show using simulated data and several datasets from the UCI database that

the proposed algorithms results in sparse and accurate classifiers. Finally some discussions are reported in Section 5.

### 1.1. *Related work*

In recent years, there has been interest in the sparsity of the resulting classifier [5, 15, 16]. There are two notions of sparsity, the so-called "primal" and "dual" sparsity [17].

Dual sparsity refers to the effect that the final classifier is a combination of only a small number of training examples. The support vector machine [2] is an example of this effect. The support vector machine (SVM) minimizes the hinge loss on the training examples, and uses the squared $\ell_2$ norm as a regularizer.

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \sum_{i=1}^{n} \max(0, 1 - y_i(\langle w, x_i\rangle + b)) + \lambda\|w\|_2^2.$$

One benefit of the $\ell_2$ norm is that one can replace it with the reproducing kernel Hilbert space norm, which allows us to prove the representer theorem. This is the basis for the kernel trick, which generalizes the SVM to non-linear classification. The representer theorem asserts that the resulting classifier is a convex combination of training points. The hinge loss promotes sparsity in the number of examples used in the final classifier, the so-called support vectors. This is because the hinge loss results in an $\ell_1$ penalty on the Lagrange multipliers corresponding to the soft constraint that the examples have to be on the correct side of the resulting hyperplane. Due to the effect of the hinge loss on the dual optimization problem, many of the dual variables are zero at optimality, resulting in few support vectors [9]. SVM has been shown to be consistent for universal kernels [18], but an improvement is still expected since the ramp loss results in better learning rates [8]. There has been recent work on further increasing the sparsity of the hinge loss by truncating it at a certain level. This has been called the ramp loss or the truncated hinge loss [5, 6]. In [7], it is shown that the truncation results in Fisher consistency of multiclass classifiers. In addition, the truncated hinge loss results in tighter learning theoretic bounds [8, 9], since it is bounded while the hinge loss is unbounded. One can also consider losses which do not result in sparsity with respect to the number of examples, such as the logistic loss, but we do not consider such losses here.

Primal sparsity refers to the effect that the resulting classifier only uses a few of the features in the data. This has been studied for the regression case in the statistics community [19], by considering different regularization (or penalty terms), $\Omega(\cdot)$. Ultimately, the most sparse representation is obtained by using the $\ell_0$ norm as a regularizer but the resulting optimization problem is combinatorial that is known to be NP-hard. Hence researchers often perform a convex relaxation of the regularizer, for example to the $\ell_1$ norm [15, 20].

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \sum_{i=1}^{n} \max(0, 1 - y_i(\langle w, x_i\rangle + b)) + \lambda\|w\|_1.$$

This $\ell_1$ style penalty was popularized by the LASSO algorithm [16]. When even

more sparsity is desired, one can consider $\ell_p$ penalties for $0 < p < 1$.

$$\min_{w,b} \sum_{i=1}^{n} \max(0, 1 - y_i(\langle w, x_i \rangle + b)) + \lambda \|w\|_p^p.$$

When $p = 0$, this results in the $\ell_0$ penalty that was recently investigated in [10]. We discuss this further in Section 2.2.

Motivated by the sparsity behaviour of the ramp loss, we investigate sparsity in terms of both the examples and features. The theoretical [19] and experimental [21] investigations on sparsity have concentrated on the regression case, whereas we study the classification problem. In the paper by [17], they study the $\ell_1$ norm with hinge loss, albeit for the more complex setting of structured output learning. We propose two new objective functions, corresponding to the ramp loss versions of the capped $\ell_1$ and $\ell_0$ SVMs [10, 11]. We also investigate the newly proposed capped $\ell_1$ regularizer [12], and present the corresponding SVM forms. In [11], an expensive bilinear programming approach was used to solve the $\ell_2$ regularization problem with ramp loss. However, the paper also considers the capped $\ell_1$ regularizer, albeit not in the classification setting. The nonconvex objectives ZeroNormSVM [10], RLSVM2 [5] and RLSVM1 [7] were also solved by DCA.

### 1.2. *DCA in machine learning*

In the current paper, motivated by the efficiency of DC programming and DCA for large scale nonconvex problems, we develop these tools for solving the new resulting optimization problems.

We note that the convex concave procedure (CCCP) for constructing discrete time dynamical systems studied in [22] is nothing else than a special case of DCA. Whereas the CCCP approach assumes differentiable objective functions, DCA handles both smooth and nonsmooth nonconvex optimization. In the last five years DCA has been successfully applied in several works in machine learning for SVMs-based feature selection [5, 7, 10, 23], for improving boosting algorithms [24], for implementing learning [25, 26], and for clustering [27, 28].

DCA has been successfully used for non-convex optimization models that combined SVM-based Feature Selection and Classification [5, 7, 10, 23]. Also, the Successive Linearization Algorithm (SLA) proposed in [29] for the feature selection concave problem is a special case of the general DCA scheme.

## 2. Loss, regularizer, and new optimization models

We are given $n$ training data points $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Recall that we consider linear classifiers with an objective function of the form in Equation (1):

$$\min_{w,b} \sum_{i=1}^{n} \ell(y_i, x_i; w, b) + \lambda \Omega(w),$$

where $\ell$ is a loss function and $\Omega(\cdot)$ is the regularizer (or penalty term). We use the hyperparameter $\lambda$ in this paper to trade off between the various losses and regularizers. In anticipation of solving the optimization problems using DCA, we also present the corresponding DC decompositions in the following.

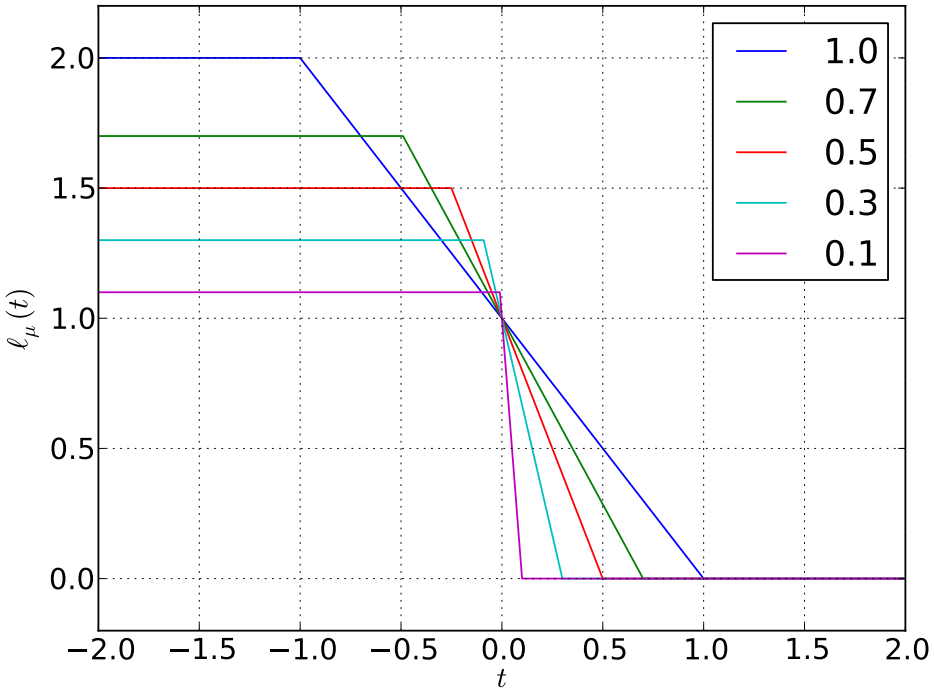


Figure 1. $\mu$-ramp loss for different values of $\mu$.

### 2.1. *$\mu$-ramp loss*

(Figure 1 should be roughly here)

Instead of the truncated hinge loss as proposed in [5], we consider a slightly different formulation. We propose a different parametrization called the $\mu$-ramp loss (shown for different values of $\mu$ in Figure 1), The $\mu$-ramp loss always upper bounds the 0-1 loss, and as $\mu \longrightarrow 0$ tends to the 0-1 loss. For larger values of $\mu$, this is similar to the truncated hinge loss. The $\mu$ ramp loss is defined as

$$\ell_\mu(t) = \begin{cases} \mu + 1 & \text{if } t < -\mu^2 \\ 1 - \frac{t}{\mu} & \text{if } -\mu^2 \leqslant t \leqslant \mu \\ 0 & \text{if } t > \mu \end{cases} \tag{2}$$

which can also be expressed as

$$\ell_\mu(t) = \frac{1}{\mu}\max(0, \mu - t) - \frac{1}{\mu}\max(0, -\mu^2 - t) \text{ for } \mu \in [0, 1]. \tag{3}$$

Reducing $\mu$ is a double edged sword. On one hand, we would expect it to increase the sparsity of the resulting classifier, since we reduce the region with non-zero gradient in the dual variables [5]. The ramp loss has also been shown to result in tighter generalization bounds [6]. However, the ramp loss with small $\mu$ has a large Lipschitz constant, which results in poor generalization error bounds [8]. For most of the experiments in Section 4, we keep $\mu = 1$, and hence are in a similar setting to previous work. We further investigate the effect of reducing the value of $\mu$ in Section 4.3.

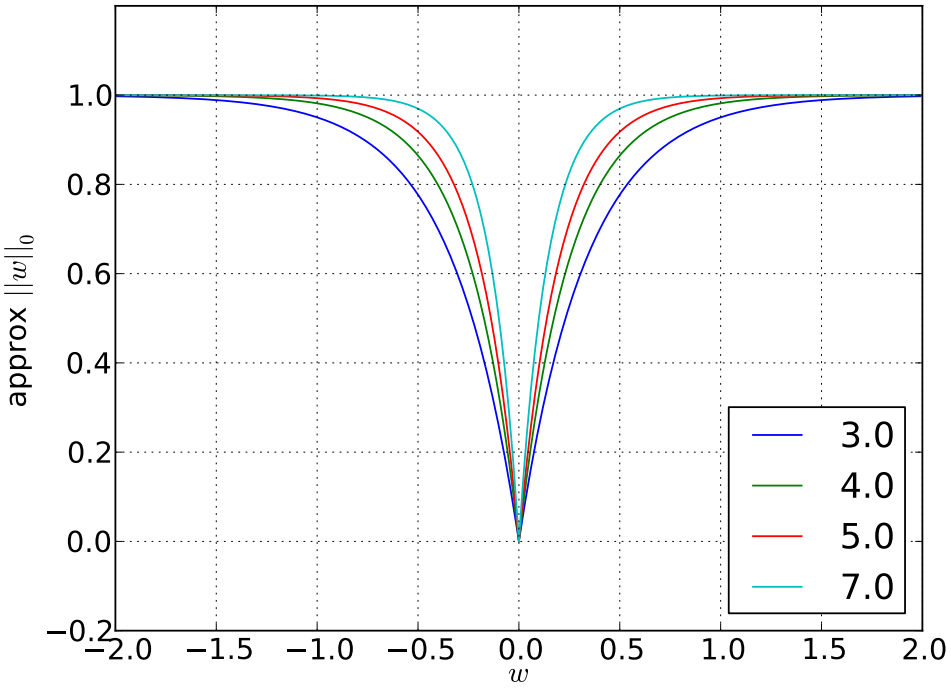


Figure 2. Approximation of the $\ell_0$ regularizer, for various values of $a$.

### 2.2. *Approximation of the $\ell_0$ regularizer*

(Figure 2 should be roughly here)

The $\ell_0$ norm results in a combinatorial optimization problem, and hence is not practical for large scale problems. We consider a smooth approximation to the $\ell_0$ norm [10]. For $\tau \in \mathbb{R}$, let $r_a(\tau)$ be the function defined for a given $a > 0$ by

$$r_a(\tau) = 1 - \exp(-a|\tau|). \tag{4}$$

The parameter $a$ controls the "steepness" of the valley. A plot of the regularizer for various values of $a$ is given in Figure 2. This can be represented as a difference of convex functions $r_a(\tau) = g_a(\tau) - h_a(\tau)$, given by

$$g_a(\tau) = a|\tau| \text{ and } h_a(\tau) = a|\tau| - 1 + \exp(-a|\tau|). \tag{5}$$

### 2.3. *The capped $\ell_1$ regularizer*

The capped $\ell_1$ regularizer has been recently proposed [11] and shown to have useful statistical properties [12]. For $\tau \in \mathbb{R}$, let $\delta_a(\tau)$ be the function defined for a given $a > 0$ by

$$\delta_a(\tau) = \min(a, |\tau|). \tag{6}$$

The parameter $a$ controls the maximum possible penalty that one pays for a large weight. This can be represented as a difference of convex functions given by

$$\delta_a(\tau) = g_a(\tau) - h_a(\tau), \text{ where } g_a(\tau) = |\tau| \text{ and } h_a(\tau) = \max(0, |\tau| - a). \tag{7}$$

### 2.4. *New optimization models*

We propose three combinations of loss and regularizer presented above and derive the resulting optimization problems. In particular: we combine $\mu$-ramp loss with the $\ell_0$ penalty (RLSVM0), derive an efficient optimization method for hinge loss with capped $\ell_1$ penalty (CapOneNormSVM), and also propose combining the $\mu$-ramp loss with capped $\ell_1$ penalty (RLSVMC1). For the sake of completeness, we also present (in the appendix) the resulting optimization problems of other five previously known cases indicated in Table 1 that we consider in the computational experiments.

#### *2.4.1. $\ell_0$ regularizer approximate with $\mu$-Ramp Loss*

The first new formulation, called the $\ell_0$ regularized ramp loss SVM (RLSVM0), consists of combining the $\mu$-ramp loss in (3) and the approximation to the $\ell_0$ norm in (4). The resulting optimization problem takes the form

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} F_1(w,b) := \sum_{i=1}^{n} \ell_\mu(y_i(\langle w, x_i\rangle + b)) + \lambda \sum_{j=1}^{d} r_a(w_j). \tag{8}$$

#### *2.4.2. Capped $\ell_1$ regularizer with hinge loss*

For the second formulation, called the capped $\ell_1$ regularized hinge loss SVM (CapOneNormSVM), we combine the hinge loss in standard SVM and the capped $\ell_1$ in (6). The resulting optimization problem is given by

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} F_2(w,b) := \sum_{i=1}^{n} \max(0, 1 - y_i(\langle w, x_i\rangle + b)) \ + \lambda \sum_{j=1}^{d} \delta_a(w_j). \tag{9}$$

We propose an efficient optimization method in Section 3.

#### *2.4.3. Capped $\ell_1$ regularizer with $\mu$-ramp loss*

The combination of $\mu$-ramp loss with capped $\ell_1$ regularizer (RLSVMC1) gives birth the third new optimization model which is described as

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} F_3(w,b) := \sum_{i=1}^{n} \ell_\mu(y_i(\langle w, x_i\rangle + b)) + \lambda \sum_{j=1}^{d} \delta_a(w_j). \tag{10}$$

The three objectives proposed have the advantage that, while non-convex, they can be expressed as a difference of two convex functions. Hence DCA can be applied on these problems.

## 3. Solution methods by Difference of Convex functions Algorithm (DCA)

DC programming and DCA have been introduced by [30] and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1993 ([13, 14] and references therein). They constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. We express the three objective functions proposed in Section 2.4 in the framework of DCA, and analyse their convergence properties here.

### 3.1. *General DC programs*

To give the reader some background on the theory of DC programming and DCA, and our motivation to use them in the present work, we briefly outline these tools at the beginning of this section. Let $\Gamma_0(\mathbb{R}^m)$ denotes the convex cone of all lower semi-continuous proper convex functions on $\mathbb{R}^m$. The generic DCA addresses a DC program which takes the form

$$\alpha = \inf\{F(z) := G(z) - H(z) : z \in \mathbb{R}^m\} \quad (P_{dc})$$

where $G, H \in \Gamma_0(\mathbb{R}^m)$. Note that if $(P_{dc})$ admits a solution, that is $\alpha$ is finite, then one can replace "inf" by "min". Convex constraints on $z$ can be expressed in the form above by using the indicator set of the constraints. Let $C$ be a nonempty closed convex set. Then, the problem

$$\inf\{F(z) := G(z) - H(z) \ : \ z \in C\}$$

can be transformed into an unconstrained DC program by using the indicator function of $C$ denoted by $\chi_C$, that is

$$\inf\{F(z) := \phi(z) - H(z) : \ z \in \mathbb{R}^m\},$$

where $\phi := G + \chi_C$ is in $\Gamma_0(\mathbb{R}^m)$.

Non smooth functions are handled (as in convex analysis) using the concept of subdifferentials. Recall that, for $\theta \in \Gamma_0(\mathbb{R}^m)$ and $z_0 \in \text{dom}\ \theta := \{z \in \mathbb{R}^m | \theta(z_0) < +\infty\}$, the subdifferential of $\theta$ at $z_0$, denoted $\partial\theta(z_0)$, is defined as

$$\partial\theta(z_0) := \{y \in \mathbb{R}^m : \theta(z) \geq \theta(z_0) + \langle z - z_0, y\rangle, \forall z \in \mathbb{R}^m\}$$

which is a closed convex set in $\mathbb{R}^m$. It generalizes the derivative in the sense that $\theta$ is differentiable at $z_0$ if and only if $\partial\theta(z_0)$ is reduced to a singleton which is exactly $\{\nabla\theta(z_0)\}$.

The necessary local optimality condition for $(P_{dc})$ is

$$\partial H(z^*) \subset \partial G(z^*). \tag{11}$$

The condition (11) is also sufficient for many important classes of DC programs, for example, for DC polyhedral programs, or when function $F$ is locally convex at $z^*$ [14]. A DC program $(P_{dc})$ is called a DC polyhedral program when either $G$ or $H$ is a polyhedral convex function (i.e., the pointwise supremum of a finite collection of affine functions). Note that a polyhedral convex function is almost everywhere differentiable, that is it is differentiable everywhere except on a set of measure zero.

A point that $z^*$ verifies the generalized Kuhn-Tucker condition

$$\partial H(z^*) \cap \partial G(z^*) \neq \emptyset \tag{12}$$

is called a critical point of $G - H$. It follows that if $H$ is polyhedral convex, then a critical point of $G - H$ is almost always a local solution to $(P_{dc})$ [14].

It is worth noting the richness of the set of DC functions on $\mathbb{R}^m$: they contain many objective functions and are closed under the operations usually considered in optimization [13].

### 3.2. *Difference of Convex functions Algorithms (DCA)*

The main idea behind DCA is to replace, at the current point $z^k$ of iteration k, the concave part $-H(z)$ with its affine majorization defined by

$$H_k(z) := H(z^k) + \left\langle z - z^k, \gamma^k \right\rangle, \quad \gamma^k \in \partial H(z^k)$$

to obtain the convex program of the form

$$\inf\{G(z) - H_k(z) : \ z \in \mathbb{R}^m\} \Longleftrightarrow \inf\{G(z) - \left\langle z, \gamma^k \right\rangle \ : z \in \mathbb{R}^m\}. \quad (P_k)$$

In fact, DCA is an iterative primal-dual subgradient method, but for simplicity we omit here the DC duality and the dual part of DCA. The generic DCA scheme is shown below. Note that the DCA is constructed from DC components $G$ and $H$ and their conjugates but not from the DC function $F$ itself. Furthermore, DCA is a descent method without line-search which has linear convergence for general DC programs.

**Algorithm 1** Generic difference of convex functions algorithm (DCA)

**Initialization:** Let $z^0 \in \mathbb{R}^m$ be an initial guess.
$k = 0$
**repeat**
$\gamma^k \in \partial H(z^k)$.
$z^{k+1} \in \operatorname{argmin}\{G(z) - \left\langle z, \gamma^k \right\rangle : z \in \mathbb{R}^m\}$.
$k = k + 1$
**until** convergence

DCA schemes have the following properties [14]:

i) the sequence $\{G(z^k) - H(z^k)\}$ is decreasing,
ii) if the optimal value $\alpha$ of problem $(\mathrm{P}_{dc})$ is finite and the infinite sequences $\{z^k\}$ *is* bounded, then every limit point $\widetilde{z}$ of the sequence $\{z^k\}$ is a critical point of $G-H$. In particular, if $H$ is polyhedral function and $H$ is differentiable at $x^*$, then $x^*$ is a local minimizer of $(\mathrm{P}_{dc})$.
iii) If $(\mathrm{P}_{dc})$ is a polyhedral DC program, then the sequence $\{z^k\}$ converges after a finite number of iterations.

Observe that a DC function has infinitely many DC decompositions and there are as many DCA as there are DC decompositions which have crucial impacts on the qualities (speed of convergence, robustness, efficiency, and globality of computed solutions) of DCA. Hence, the solution of a nonconvex program by DCA must be composed of two stages: the search of an appropriate DC decomposition and that of a good initial point. In the following subsections, we provide the details of the corresponding decompositions and initialization points for the objective functions in Section 2.4.

### 3.3. *DCA for solving RLSVM0*

Consider now the optimization problem (8), where $\ell_\mu$ is the $\mu$-ramp loss and $r_a$ is the approximate $\ell_0$ regularizer.

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} F_1(w,b) := \sum_{i=1}^{n} \ell_\mu(y_i(\langle w, x_i\rangle + b)) + \lambda \sum_{j=1}^{d} r_a(w_j).$$

A natural DC decomposition of $F_1$ can be derived from (3) and (5) as follows:

$$F_1(w,b) = G(w,b) - H(w,b) \tag{13}$$

where

$$G(w,b) = \frac{1}{\mu}\sum_{i=1}^{n} \max(0, \mu - y_i(\langle w, x_i\rangle + b)) + \lambda a\|w\|_1, \tag{14}$$

and

$$H(w,b) = \frac{1}{\mu}\sum_{i=1}^{n} \max(0, -\mu^2 - y_i(\langle w, x_i\rangle + b)) + \lambda \sum_{j=1}^{d} a|w_j| - 1 + \exp(-a|w_j|). \tag{15}$$

The problem (8) can be now written in the form of DC program:

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} G(w,b) - H(w,b). \tag{16}$$

Observe that $G(w,b)$ is the same as a margin rescaled SVM with $\ell_1$ regularization. Moreover, it is worth noting that $G$ is a convex polyhedral function, consequently (16) is a DC polyhedral program. Therefore, as will be seen later, DCA applied to (16) converges finitely, and it requires solving one linear program at each iteration. For designing DCA applied to (16), firstly, we have to compute a subgradient of the function $H$. Consider the first term of $H$ in Equation (15),

$$H^1(w,b) := \frac{1}{\mu}\sum_{i=1}^{n} h_i^1(w,b), \text{ with } h_i^1(w,b) := \max(0, -\mu^2 - y_i(\langle w, x_i\rangle + b)).$$

We have

$$\partial h_i^1(w,b) = \partial_w h_i^1(w,b) \times \partial_b h_i^1(w,b)$$

where [31]:

$$\mathbb{R}^d \supset \frac{\partial h_i^1(w,b)}{\partial w} = \begin{cases} \{-y_i x_i\} & \text{for} \quad y_i(\langle w, x_i\rangle + b) < -\mu^2 \\ \{0\} \in \mathbb{R}^d & \text{for} \quad y_i(\langle w, x_i\rangle + b) > -\mu^2 \\ \text{co}\,\{0, -y_i x_i\} & \text{for} \quad y_i(\langle w, x_i\rangle + b) = -\mu^2 \end{cases} \tag{17}$$

and

$$\mathbb{R} \supset \frac{\partial h_i^1(w,b)}{\partial b} = \begin{cases} \{-y_i\} & \text{for} \quad y_i(\langle w, x_i\rangle + b) < -\mu^2 \\ 0 & \text{for} \quad y_i(\langle w, x_i\rangle + b) > -\mu^2 \\ \text{co}\,\{0, -y_i\} & \text{for} \quad y_i(\langle w, x_i\rangle + b) = -\mu^2. \end{cases} \tag{18}$$

Here "co" stands for the convex hull.

Observe that $h_i^1(w,b)$ is differentiable everywhere, except for $y_i(\langle w, x_i\rangle + b) = -\mu^2$. Since the DCA only needs an element from the subdifferential, we choose (arbitrarily) that

$$0 \in \mathbb{R}^{d+1} \text{ as a subgradient of } \partial h_i^1(w,b) \text{ if } y_i(\langle w, x_i\rangle + b) = -\mu^2. \tag{19}$$

Finally, the subgradient of $H^1(w,b)$ corresponding to this choice is written as

$$\gamma \in \partial H^1(w,b) \Longleftarrow \gamma = \frac{1}{\mu}\sum_{i=1}^{n}(s_{wi}, s_{bi}), \text{ with } s_{wi} \in \frac{\partial h_i^1(w,b)}{\partial w},\ s_{bi} \in \frac{\partial h_i^1(w,b)}{\partial b} \tag{20}$$

being defined by (17), (18) and (19).

Consider now the second term of $H$ in Equation (15),

$$H^2(w,b) := \lambda \sum_{i=1}^{n}\sum_{j=1}^{d} a|w_j| - 1 + \exp(-a|w_j|).$$

It has been shown in [10] that $H^2(w,b)$ is differentiable everywhere and $\nabla H^2(w,b) = (\lambda v, 0)$ with

$$v_j = \begin{cases} a(1-\exp(-aw_j)) & \text{if } w_j \geq 0 \\ -a(1-\exp(aw_j)) & \text{if } w_j < 0 \end{cases}, j = 1, \ldots, d. \tag{21}$$

Finally, a subgradient $(\eta, \kappa)$ of the function $H$ (which is in fact differentiable everywhere except for some that points verifying $y_i(\langle w, x_i\rangle + b) = -\mu^2$) is computed as

$$\mathbb{R}^d \times \mathbb{R} \supset \partial H(w,b) \ni (\eta, \kappa) = \frac{1}{\mu}\sum_{i=1}^{n}(s_{wi}, s_{bi}) + (\lambda v, 0) \tag{22}$$

with $s_{wi}, s_{bi}$ being defined by (17), (18) and (19) and $v = (v_j)$ being given in (21).

According to the generic DCA scheme, at each iteration $k$, we have to compute $(\eta^k, \kappa^k) \in \partial H(w^k, b^k)$ and then solve the convex program of the form $(P_k)$ (Section 3.1),

$$\min\left\{G(w,b) - \left\langle (\eta^k, \kappa^k), (w,b)\right\rangle : (w,b) \in \mathbb{R}^{d+1}\right\}. \tag{23}$$

Since $G(w,b) = \frac{1}{\mu}\sum_{i=1}^{n}\max(0, \mu - y_i(\langle w, x_i\rangle + b)) + \lambda a\|w\|_1$ is a polyhedral convex

function, the problem(23) is equivalent to the following linear program:

$$\begin{array}{ll} \min\limits_{w,b,\xi,\varsigma} & \frac{1}{\mu}\sum\limits_{i=1}^{n}\xi_i + \lambda a\sum\limits_{j=1}^{d}\varsigma_j - \left\langle \eta^k, w\right\rangle - \kappa^k b \\ \text{subject to} & \xi_i \geqslant \mu - y_i(\langle w, x_i\rangle + b) \\ & -\varsigma_j \leqslant w_j \leqslant \varsigma_j, \text{ for } j = 1,\ldots,d \\ & \xi_i, \varsigma_j \geqslant 0, \text{for } i = 1,\ldots,n, j = 1,\ldots,d. \end{array} \tag{24}$$

As mentioned earlier, a good initialization point is important for finding a good solution. In the setting of this paper, it is natural to use the linear program corresponding to the convex part of the DC function. This is fortuitously the convex program OneNormSVM (Appendix A.2), hence implying that the solution obtained by the proposed DCA is at least as good as that obtained by this well studied method [15]. The DCA for solving RLSVM0 is shown in Algorithm 2.

**Algorithm 2** RLSVM0

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard 1-norm SVM solution.
Set hyperparameter $\lambda$ and $c$.
**repeat**
Compute $(\eta^k, \kappa^k) \in \partial H(w^k, b^k)$.
Solve the linear program (24) to obtain $w^{k+1}, b^{k+1}$,
**until** $F_1(w^{k+1}, b^{k+1}) = F_1(w^k, b^k)$ (Equation 13)

### 3.4. *DCA for solving CapOneNormSVM*

We consider the capped $\ell_1$ regularizer (9) proposed in [11] with the hinge loss, and derive a DCA to solve it. Note that in [11], the capped $\ell_1$ regularizer was not used for classification, but a similar problem was solved using bilinear programming. The DC decomposition of $\delta_a(\tau)$ given in (7) implies the following DC decomposition:

$$F_2(w, b) = \Phi(w, b) - \Psi(w, b) \tag{25}$$

where

$$\Phi(w,b) = \sum_{i=1}^{n}\max(0, 1 - y_i(\langle w, x_i\rangle + b)) + \lambda\|w\|_1, \quad \Psi(w,b) = \lambda\sum_{j=1}^{d}\max(0, |w_j| - a).$$

This is a polyhedral DC program where both $\Phi$ and $\Psi$ are convex polyhedral functions. Similar to the computation of subgradient of $H$ we have

$$\partial\Psi(w, b) \ni (\lambda v, 0) \tag{26}$$

where

$$v_j = \begin{cases} \text{sign}(w_j) & \text{for} \quad |w_j| \geqslant a \\ 0 & \text{for} \quad |w_j| < a \end{cases}, \; j = 1,\ldots,d. \tag{27}$$

Let $(\lambda v^k, 0) \in \partial\Psi(w^k, b^k)$. Like (23), the convex program

$$\min\left\{\Phi(w,b) - \left\langle (\lambda v^k, 0), (w,b)\right\rangle : (w,b) \in \mathbb{R}^{d+1}\right\}$$

is equivalent to the following linear program:

$$\begin{array}{ll} \min\limits_{w,b,\xi,\varsigma} & \sum\limits_{i=1}^{n}\xi_i + \lambda\sum\limits_{j=1}^{d}\varsigma_j - \lambda\left\langle v^k, w\right\rangle \\ \text{subject to} & \xi_i \geqslant 1 - y_i(\langle w, x_i\rangle + b) \\ & -\varsigma_j \leqslant w_j \leqslant \varsigma_j, \text{ for } j = 1,\ldots,d \\ & \xi_i, \varsigma_j \geqslant 0, \text{for } i = 1,\ldots,n, j = 1,\ldots,d. \end{array} \quad . \tag{28}$$

Hence the DCA applied to (9) is described in Algorithm 3.

**Algorithm 3** CapOneNormSVM

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard 1-norm SVM solution.
Set hyperparameter $\lambda$ and $a$.
**repeat**
Compute $v^k$ via (27).
Solve the linear program (28) to obtain $w^{k+1}, b^{k+1}$,
**until** $F_2(w^{k+1}, b^{k+1}) = F_2(w^k, b^k)$ (Equation 25)

### 3.5. *DCA for solving RLSVMC1*

Using the same technique as in the previous two DC formulations, using (3) and (7) we obtain the next DC formulation of the problem (10):

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} F_3(w,b) := \Xi(w,b) - \Upsilon(w,b) \tag{29}$$

where

$$\Xi(w,b) = \frac{1}{\mu}\sum_{i=1}^{n}\max(0, \mu - y_i(\langle w, x_i\rangle + b)) + \lambda\|w\|_1, \quad \Upsilon(w,b) = H^1(w,b) + \Psi(w,b).$$

Hence (29) is a polyhedral DC program where both $\Xi$ and $\Upsilon$ are convex polyhedral functions. Using (20) and (26) we can determine a subgradient of $\Upsilon(w,b)$ as follows

$$\partial\Upsilon(w,b) \ni (\vartheta, \sigma) := \left(\left(\frac{1}{\mu}\sum_{i=1}^{n} s_{wi}\right) + \lambda v, \frac{1}{\mu}\sum_{i=1}^{n} s_{bi}\right) \tag{30}$$

where $s_{wi} \in \frac{\partial h_i^1(w,b)}{\partial w}$, $s_{bi} \in \frac{\partial h_i^1(w,b)}{\partial b}$ are defined in (17), (18) and (19), and $v$ is computed by (27). Let $(\vartheta^k, \sigma^k) \in \partial\Upsilon(w^k, b^k)$. Similar to (23), the convex program

$$\min\left\{\Xi(w,b) - \left\langle (\vartheta^k, \sigma^k), (w,b)\right\rangle : (w,b) \in \mathbb{R}^{d+1}\right\}$$

is equivalent to the following linear program:

$$\begin{array}{ll} \min\limits_{w,b,\xi,\varsigma} & \frac{1}{\mu}\sum\limits_{i=1}^{n}\xi_i + \lambda\sum\limits_{j=1}^{d}\varsigma_j - \left\langle \vartheta^k, w\right\rangle - \sigma^k b \\ \text{subject to} & \xi_i \geqslant \mu - y_i(\langle w, x_i\rangle + b) \\ & -\varsigma_j \leqslant w_j \leqslant \varsigma_j, \text{ for } j = 1,\ldots,d \\ & \xi_i, \varsigma_j \geqslant 0, \text{for } i = 1,\ldots,n, j = 1,\ldots,d. \end{array} . \tag{31}$$

**Algorithm 4** RLSVMC1

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard 1-norm SVM solution.
Set hyperparameter $\lambda$ and $c$.
**repeat**
Compute $(\vartheta^k, \sigma^k) \in \partial\Upsilon(w^k, b^k)$ via (30)
Solve the linear program (31) to obtain $w^{k+1}, b^{k+1}$,
**until** $F_3(w^{k+1}, b^{k+1}) = F_3(w^k, b^k)$ (Equation 29)

### 3.6. *Convergence*

The three DCA schemes developed above enjoy interesting convergence properties. For notational convenience, we denote by $F$ the objective function of the resulting optimization problems, that is $F \in \{F_1, F_2, F_3\}$.

THEOREM 3.1 Convergence properties of proposed algorithms
*For Algorithms 2, 3 and 4, we have the following:*

*(I) DCA generates the sequence* $\{(w^k, b^k)\}$ *such that the corresponding sequence* $\{F(w^k, b^k)\}$ *is monotonously decreasing.*
*(II) The sequence* $\{(w^k, b^k)\}$ *converges to* $(w^*, b^*)$ *after a finite number of iterations.*
*(III) The point* $(w^*, b^*)$ *is a critical point of* $F$.
*(IV) The point* $(w^*, b^*)$ *is almost always a local minimizer of the corresponding optimization problem (8), (9) or (10). More precisely*
*(a) In* **CapOneNormSVM,** *if*

$$w_j^* \neq a \ \forall j = 1,\ldots,d \tag{32}$$

*then* $(w^*, b^*)$ *is a local minimizer of (9).*
*(b) In* **RLSVMC1**, *if*

$$w_j^* \neq a \ \forall j = 1,\ldots,d \text{ and } y_i(\langle w, x_i\rangle + b) \neq \mu^2 \ \forall i = 1,\ldots,n \tag{33}$$

*then* $(w^*, b^*)$ *is a local minimizer of (10).*

*Proof* (I) and (III) are direct consequences of the convergence properties of general DC programs while (II) is a convergence property of a DC polyhedral program. Moreover, observing that the second DC component of (9) and (10), say $\Psi$ and $\Upsilon$ are convex polyhedral functions, and if the condition (32) (resp. (33)) holds, then $\Psi$ (resp. $\Upsilon$) is differentiable at $(w^*, b^*)$, and using the DCA's convergence property ii) mentioned in Section 3.2 we deduce (IV.a) and (IV.b). 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 $(w^*, b^*)$ is almost always a local minimizer of (9) and/or (10). Likewise, by considering the dual problem of (8) in which the second DC component is convex polyhedral, we can prove that **RLSVM0** converges almost always to a local minimizer of (8). ■

In fact, for the ramp loss, detecting convergence can simply be done by checking whether any new points fall into the margin region. Since the considered problem is non-convex, we may end up in a local optimum. Knowing that DCA with a good starting point can provide a global minimizer, we initialize our methods with the solution of a "close" convex problem, which is $\ell_1$ regularized SVM.

## 4. Computational experiments

We implemented the above optimization problems using Python[1] and CVXOPT[2]. The optimization code and the software for reproducing the experiment is available at `http://www.inf.ethz.ch/personal/cong/optwok.html`

We used the following experimental settings for all our experiments. We used a 70%/30% training-test set split, and the results shown are mean and standard deviation of 20 random repetitions. To select the regularization parameter $\lambda$, we used five fold cross validation to choose from the set $\lambda$={0.1, 0.2, 0.3, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0}. The parameters $a$ and $\mu$ were fixed for all experiments to $a = 4$ for the $\ell_0$ regularizer, and $a = 1$ for the capped $\ell_1$ regularizer, and $\mu = 1$ (except in Section 4.3). The DCA iterations are stopped when the total change in parameters are less than $\varepsilon = 10^{-3}$. Note that apart from Equation (4), all the other DCA have discrete changes, and hence the stopping criteria is effectively exact (Section 3.6).

To compute the fraction of features used, we used the threshold $10^{-6}$ on the weight vector $w$, and divided the number of "non-zero" weights with the number of dimensions. Following [5], we detect the number of support vectors by applying the classifier on the training data and counted the number of examples in the sloped region of the loss, again using a small threshold. For the hinge loss, this includes all points with the signed output less than $1 + 10^{-2}$, and for the ramp loss this includes all points in the interval $[-\mu^2 - 10^{-2}, \mu + 10^{-2}]$.

We compare our new proposed DCAs (Section 3.2) with the three DCA schemes developed recently in [5], [7] and [10], and also consider the two standard convex objective functions (refer to Table 1, Section 2.4 and the appendix). We are interested in the efficiency of different DCA schemes for this class of problems dealing with sparsity of the resulting classifier. Recall that we are considering two types of sparsity: the number of support vectors and the number of features with non-zero weight.

### 4.1. *Simulated data*

We generated some data from two 10 dimensional Gaussian distributions with variance 1.5, centered around the unit positive and negative vectors. In addition, we generated noise by sampling uniformly in the other dimensions. We investigated $[0, 10, 100, 200]$ noise dimensions. For normalization, we scaled the resulting dataset by 0.2. The following results are based on 20 random permutations of the data,

[1] `http://www.python.org`
[2] `http://abel.ee.ucla.edu/cvxopt/`

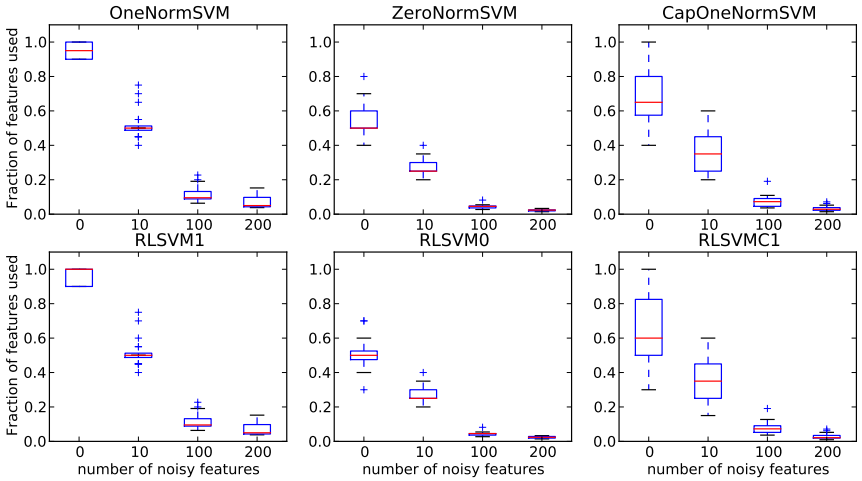


Figure 3. The fraction of features used when increasing noise dimensions. The $\ell_2$ regularized classifiers always use all features, and are not shown. The fraction of true features are $[1.0, 0.5, 0.1, 0.05]$ respectively. The classifiers with $\ell_0$ and capped $\ell_1$ regularization use fewer features.

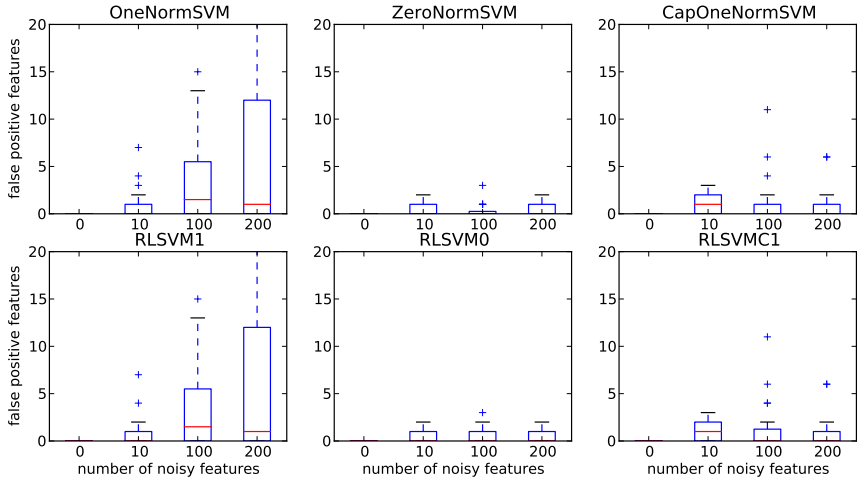


Figure 4. The fraction of false positive features used when increasing noise dimensions. The $\ell_2$ regularized classifiers always use all features, and are not shown. $\ell_1$ regularized classifiers discover an increasing number of "noise" features, whereas the other two regularizers do not.

where in each permutation 70 examples were used in training and the remaining 30 were used for testing.

### *4.1.1. Feature selection*

(Figures 3, 4, 5 and 6 should be roughly here)

In this section, we investigate the ability of the algorithms in finding the dimensions corresponding to the Gaussian distributions (the "true" features). We consider elements $w_j > 10^{-6}$ of the weight vector to be nonzero, and deem the corresponding feature to be selected. Since we generated the data, we can compare the discovered features with the ground truth.

Recall that the $\ell_2$ regularized classifiers always use all features, hence we investigate the remaining classifiers. As can be seen in Figure 3, the resulting classifiers are sparse in the number of selected features in the presence of noise dimensions. Note also that the number of chosen features is close to the right fraction of true features. Features which are given zero weight are called false negative features,

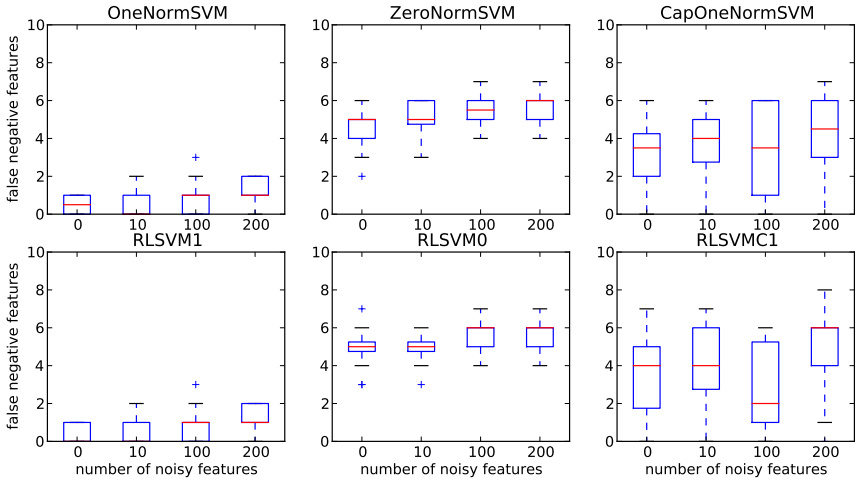


Figure 5. The fraction of false negative features used when increasing noise dimensions. The $\ell_2$ regularized classifiers always use all features, and are not shown. The classifiers with $\ell_0$ and capped $\ell_1$ regularization ignore a portion of the "informative" features, however with little loss of accuracy (see Figure 6).

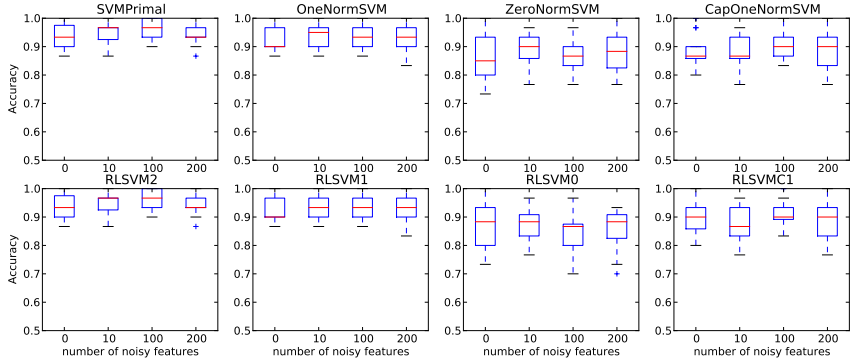


Figure 6. The accuracy with respect to increasing noise dimensions. The top row shows the performance of classifiers with hinge loss, and the bottom classifiers with ramp loss. The classifiers with sparsity inducing regularizers suffer a small decrease in accuracy compared to the $\ell_2$ regularized classifier.

and "noise" features which are given non-zero weight are called false positive features. Figure 4, and Figure 5 show that the $\ell_0$ norm and the capped $\ell_1$ norm are better at filtering out false features, but at the cost of missing some true features. It is known from the feature selection literature that not all informative features are needed to achieve good accuracy. It may be that some features are redundant (although informative) given the other available features [32]. From Figure 6, we see that when comparing classifiers with the same loss function, the sparsity does not significantly reduce the accuracy of the classifier.

### *4.1.2. Robustness against label noise*

(Figure 7 should be roughly here)

One benefit of the ramp loss is that it "gives up" on examples which are too far on the wrong side of the hyperplane, for example when a particular example is falsely labeled. In this section, we show that the increase in the number of support vectors is reduced.

To test the intuition that the ramp loss is effective against random classification noise, we generated data as above (with no noise dimensions), and randomly changed the label for certain examples in the data set. We tested the fractions $[0.0, 0.1, 0.2, 0.3]$, that is we randomly flipped the labels of 0, 10, 20 and 30 of the

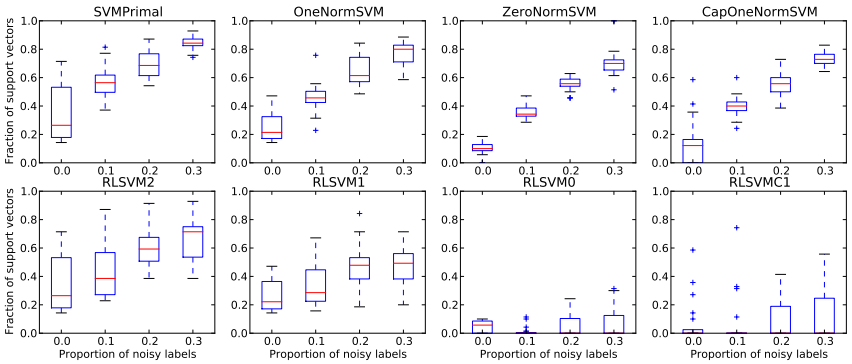


Figure 7. The fraction of support vectors when increasing random classification noise. The classifiers with ramp loss has a smaller increase in the number of support vectors as the random classification noise is increased.

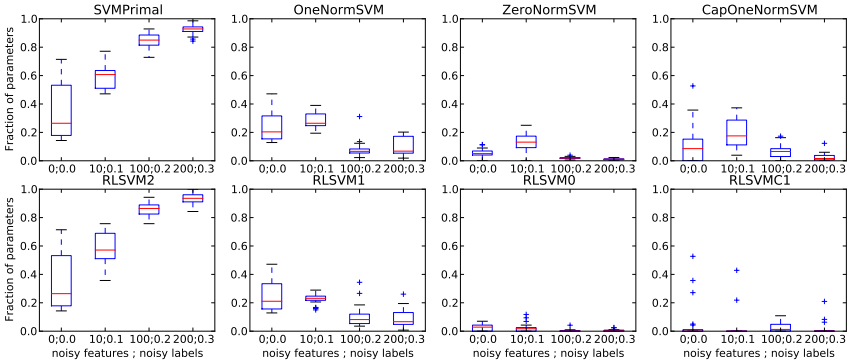


Figure 8. The fraction of parameters (support vectors and features) when increasing both the noise dimensions and random classification noise. RLSVM0 and RLSVMC1 use fewer parameters due to simultaneous sparsity in features and examples.

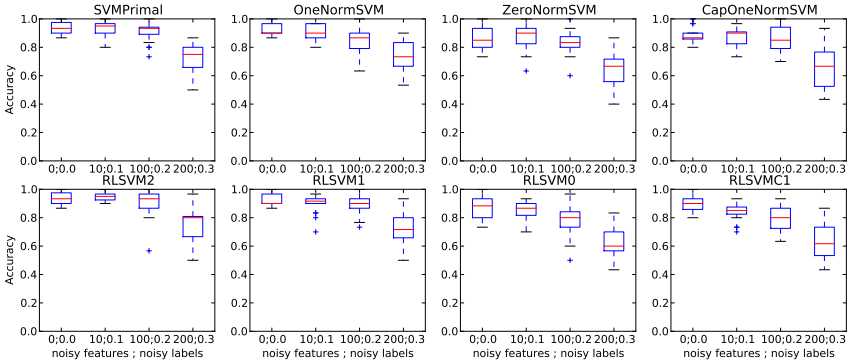


Figure 9. The accuracy with respect to increasing noise dimensions and label noise. The top row shows the performance of classifiers with hinge loss, and the bottom classifiers with ramp loss. Each element on the x-axis is a dataset labeled by two arguments (number of noisy features; fraction of noisy labels). Recall that the expected accuracy (due to random classification noise) is [1.0, 0.9, 0.8, 0.7] respectively. There is little decrease in performance in the sparsity inducing classifiers (refer to Figure 8).

100 examples.

The effect of increasing the number of falsely labeled examples in training on the number of support vectors are shown in Figure 7. Observe that the classifiers with ramp loss has a smaller increase in the number of support vectors as the random classification noise is increased.

(Figure 8 and 9 should be roughly here)

The total effect of simultaneously varying both the number of noise dimensions

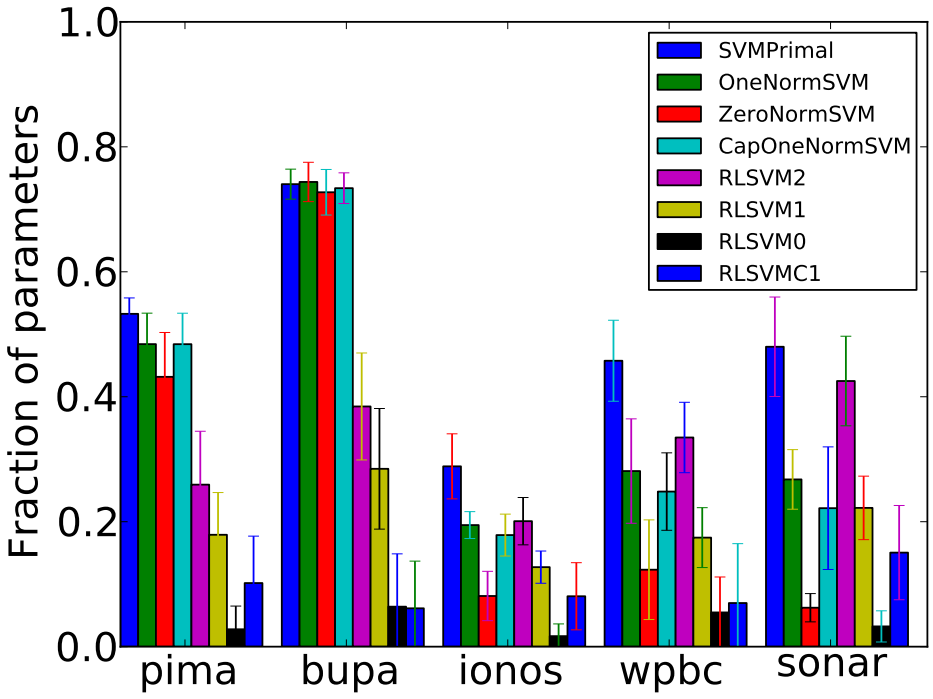


Figure 10. Effect of the algorithm used on the total fraction of parameters.

and the rate of random classification noise is shown in Figure 8. Here the fraction of parameters is given by

$$\frac{\text{number of features} \ \times \ \text{number of support vectors}}{\text{number of dimensions} \ \times \ \text{number of examples}}.$$

Observe that the sparsity pattern translates to a big savings in the total number of parameters, without a significant loss of accuracy (Figure 9).

### 4.2. *Sparsity in UCI data*

(Figure 10 should be about here.)

Using several UCI datasets, we investigated the fraction of features used and the fraction of support vectors for the six classifiers. All the classifiers had comparable accuracies. As an indication, the performance of the ramp loss SVM with $\ell_2$ regularization is shown in Table 2. Observe that the ramp loss classifiers use consistently less support vectors than their corresponding hinge loss counterparts, as expected. Also, except for the bupa data, the number of features used decreases as we go from $\ell_2$ to $\ell_1$ to $\ell_0$ norm regularization. This is also in line with the intuition that the lower norms should be sparser. The total number of parameters used for various algorithms, as shown in Figure 10, shows two groups of data. The first group which includes pima and bupa, sees a continual decrease in the total fraction of parameters. We conjecture that this is due to random classification noise in the datasets, from which the ramp loss is less susceptible. In the second group which includes ionosphere, wpbc and sonar, the savings in the total number of parameters using the ramp loss is smaller.

| | $\mu$=1.0 | | $\mu$ chosen by cross validation | | |
|---|---|---|---|---|---|
| Data | accuracy | frac sv | accuracy | frac sv | $\mu$ |
| wpbc | 0.79 ± 0.05 | 0.33 ± 0.05 | 0.78 ± 0.05 | 0.24 ± 0.06 | 0.61 ± 0.27 |
| pima | 0.76 ± 0.02 | 0.22 ± 0.11 | 0.76 ± 0.03 | 0.10 ± 0.10 | 0.49 ± 0.29 |
| ionosphere | 0.86 ± 0.03 | 0.22 ± 0.05 | 0.86 ± 0.03 | 0.17 ± 0.05 | 0.60 ± 0.29 |
| sonar | 0.74 ± 0.04 | 0.45 ± 0.06 | 0.75 ± 0.05 | 0.38 ± 0.09 | 0.68 ± 0.35 |
| bupa | 0.69 ± 0.05 | 0.41 ± 0.11 | 0.68 ± 0.03 | 0.21 ± 0.11 | 0.57 ± 0.27 |

Table 2. Reducing the number of support vectors does not affect accuracy. The results on the left are computed with $\mu = 1.0$ whereas the results on the right are obtained using cross validation. The chosen value of $\mu$ for each dataset is shown in the rightmost column. Note that for each dataset the classifiers have similar accuracies, but the number of support vectors decrease.

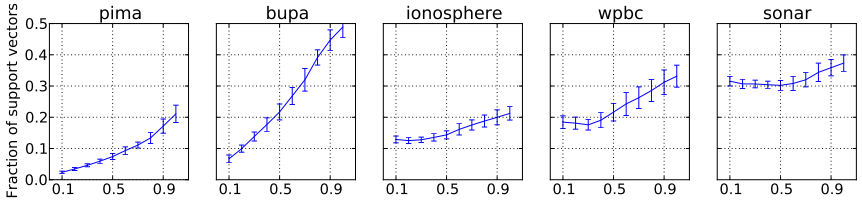


Figure 11. The effect of $\mu$ on the number of support vectors. The classifiers have similar accuracies.

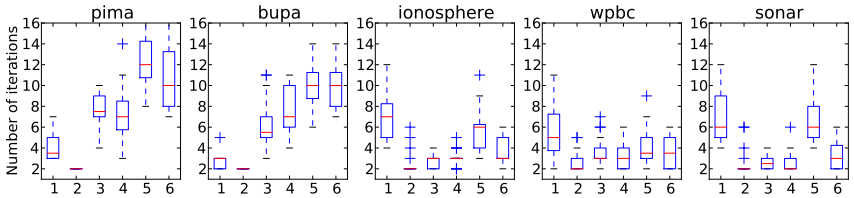


Figure 12. Number of iterations till convergence. The algorithms (1-6) are 1:ZeroNormSVM, 2:CapOneNormSVM, 3:RLSVM2, 4:RLSVM1, 5:RLSVM0, 6:RLSVMC1.

### 4.3. *Approximating 0-1 loss*

(Figure 11 should be about here.)

In this section, we investigate the effect of the parameter $\mu$. Using 5 fold cross validation, we search in for the best $\mu$ from the set $\{0.1, 0.2, \ldots, 1.0\}$. Recall that changing the value of $\mu$ affects the margin region, and as $\mu \longrightarrow 0$ the ramp loss tends to the 0-1 loss. As expected and shown in Figure 11, the number of support vectors decrease as $\mu$ decreases. Surprisingly, all the classifiers had comparable accuracies for the range $\mu$=$\{0.1, 0.2, \ldots, 1.0\}$. From Table 2, we see that by performing cross validation to find the optimal value of $\mu$, we attain similar levels of accuracy with fewer support vectors.

### 4.4. *Computational effort*

(Figure 12, 13, 14 should be about here)

All the DC methods converge within 16 iterations, as shown in Figure 12. For ionosphere and sonar, the ramp loss SVMs converge within a few iterations. Since we require an extra iteration to check whether the weights have changed, this means that for some of the datasets, the initial solution obtained using hinge loss is already optimal. However, the fraction of support vectors (for the ramp loss

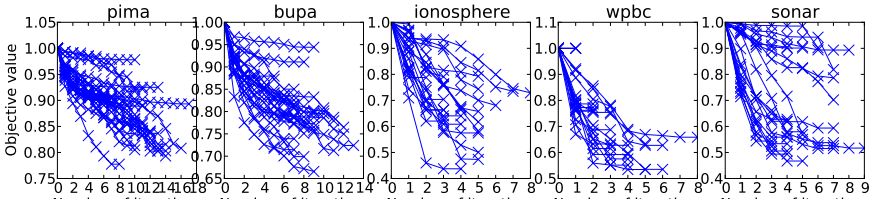


Figure 13. The normalized objective value for RLSVM0. To show the results on the same scale, the presented objective value is divided by the initial objective value.

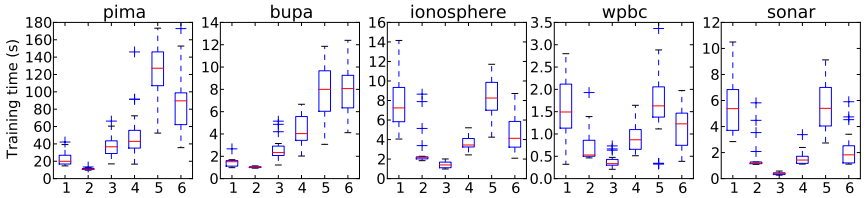


Figure 14. Training time in seconds. The algorithms (1-6) are 1:ZeroNormSVM, 2:CapOneNormSVM, 3:RLSVM2, 4:RLSVM1, 5:RLSVM0, 6:RLSVMC1.

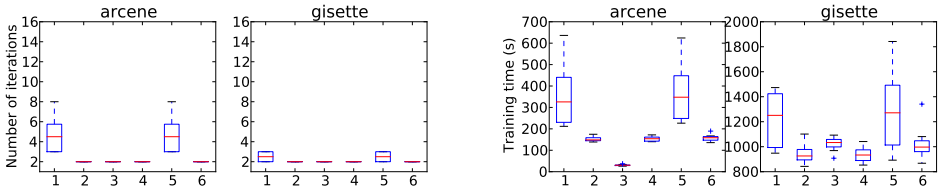


Figure 15. Larger datasets: (left) Number of iterations till convergence; (right) Training time in seconds. The algorithms (1-6) are 1:ZeroNormSVM, 2:CapOneNormSVM, 3:RLSVM2, 4:RLSVM1, 5:RLSVM0, 6:RLSVMC1.

classifiers), are further reduced by the following iterations.

The number of iterations are especially small for the non-convex regularizers when combined with hinge loss. This indicates that one can achieve feature sparsity without too much extra computation (refer to Figure 12). The objective value decreases with each iteration, which is shown in Figure 13 for RLSVM0. This shows that DCA decreases the objective value up to 50% for the considered datasets.

To show that the number of iterations till convergence does not increase significantly for larger datasets, we use two datasets from the feature selection challenge (Appendix A.7). As can be seen from Figure 15, the increase in computation time is due to the limitations of the convex optimization solver, since the number of iterations required remains small.

Our current implementation does not exploit the structure of the problems at all. It uses a general interior point package for convex optimization, CVXOPT, which does not take the constraint structure into account. However, the naive implementation only has training times of several seconds for all classifiers except for the RLSVM0 which takes several minutes on a modern desktop computer (Figure 14). The training time is dominated by the convex optimization solver. Standard tools from SVM optimization such as SMO can be brought to bear, further improving efficiency. The $\ell_1$ norm optimization problems had twice the number of variables $w$ and $\eta$, which caused it to scale poorly when the number of dimensions are very high. However, we believe that less naive implementations, such as proposed by [20, 33] would alleviate this problem.

## 5. Discussion

In this paper, we have proposed a robust continuous nonconvex optimization approach based on DC programming and DCA for the combined feature selection and SVMs. We study the effect of different losses and regularizers on the sparsity of the classifier. The two loss functions we consider are the hinge loss and the ramp loss, and the four regularizers are the $\ell_2$ and $\ell_1$ norms, and the $\ell_0$ and capped $\ell_1$ functions. The results show that we can expect savings in the number of features and the number of support vectors when we use sparsity inducing norms and losses. The setting we use is very general and the losses can be extended to the structured output setting.

One drawback of this approach is that some of the sparsity inducing functions are non-convex. However, DCA has been broadly used in other fields [14] and here we show that it is also a useful tool for optimizing non-convex objectives in machine learning. Since we use either the standard SVM or the OneNormSVM as an initial point for the weights, the resulting local optimum would be at least as good as the convex counterparts. Hence at the cost of slightly more computation time, we can achieve a sparser solution. Using an appropriate approximation function of zero-norm, we have obtained a DC polyhedral program. Our suitable DC decomposition leads to a successive linear programming algorithm that converges after a finite number of iterations to a critical point of the combined feature selection and SVMs objective function. The computational results shows that the proposed DCA performed well for all the datasets. In contrary with almost existing methods, DCA seems to do well on sparse datasets, as seen in the second experiment. Moreover, since our proposed methods involve a small number of iterations of solving a linear programming problem, it is suitable for all problems in this well studied class. Therefore, DCA is a good candidate for the combined feature selection and SVMs applications.

Note that the $\ell_0$ approximation (Equation (4)) and the capped $\ell_1$ (Equation (6)) are truncated regularizers. Unlike the other regularizers which are unbounded, the $\ell_0$ approximation is bounded above by 1. Recent work [12] has shown that such truncation of the regularizer also reduces the bias in the feature selection properties of the classifier.

We have glibly called the examples on the slope of the loss support vectors. In principle, we can really only call the examples corresponding to the $\ell_2$ regularizer support vectors. Unfortunately, for the $\ell_1$ and $\ell_0$ regularizer, we cannot use the representer theorem, and hence the notion of support vectors in these norms is not theoretically founded. Further research in this direction is required.

In summary, our empirical results show that one can reduce the number of parameters used by a classifier by considering non-convex objective functions. Additional benefits include good feature selection properties in the presence of noise dimensions, and also robustness against random classification noise.

## Appendix A. Optimization problems corresponding to Table 1

For completeness, we detail the optimization problems corresponding to related work. In the following definitions, we let $s_{wi} \in \frac{\partial h_\mu(w,b)}{\partial w}$, $s_{bi} \in \frac{\partial h_\mu(w,b)}{\partial b}$, and $u_j \in \frac{\partial h_a(w)}{\partial w_j}$. To simplify notation in the ramp loss formulations, we define $s_w = \sum_{i=1}^n s_{wi}$ and $s_b = \sum_{i=1}^n s_{bi}$.

### A.1. $\ell_2$ with hinge loss

This is the standard SVM [2].

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \sum\limits_{i=1}^n \xi_i + \lambda \langle w, w \rangle \\ \text{subject to} & \xi_i \geqslant 1 - y_i(\langle w, x_i \rangle + b) \\ & \xi_i \geqslant 0. \end{array}$$

### A.2. $\ell_1$ with hinge loss

This results in the one norm SVM [15].

$$\begin{array}{ll} \min\limits_{w,b,\xi,\eta} & \sum\limits_{i=1}^n \xi_i + \lambda \sum\limits_{j=1}^d \eta_j \\ \text{subject to} & \xi_i \geqslant 1 - y_i(\langle w, x_i \rangle + b) \\ & -\eta_j \leqslant w_j \leqslant \eta_j \\ & \xi_i, \eta_j \geqslant 0. \end{array}$$

### A.3. $\ell_0$ with hinge loss

This was recently proposed in [10]. The objective function is given by

$$F_{A3}(w,b) = \sum_{i=1}^n \max(0, 1 - y_i(\langle w, x_i \rangle + b)) + \lambda \sum_{j=1}^d r_a(w_j).$$

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard 1-norm SVM solution.

Set hyperparameter $\lambda$ and $a$.

**repeat**

Compute $u_j \in \frac{\partial h_a(w^k)}{\partial w_j}$.

Solve the linear program to obtain $w^{k+1}, b^{k+1}$,

$$\begin{array}{ll} \min\limits_{w,b,\xi,\eta} & \sum\limits_{i=1}^n \xi_i + \lambda a \sum\limits_{j=1}^d \eta_j - \lambda \langle u, w \rangle \\ \text{subject to} & \xi_i \geqslant 1 - y_i(\langle w, x_i \rangle + b) \\ & -\eta_j \leqslant w_j \leqslant \eta_j \\ & \xi_i, \eta_j \geqslant 0. \end{array}$$

**until** $F_{A3}(w^{k+1}, b^{k+1}) = F_{A3}(w^k, b^k)$

### A.4. $\ell_2$ *with ramp loss*

This was recently proposed in [5]. The objective function is given by

$$F_{A4}(w,b) = \sum_{i=1}^{n} \ell_\mu(y_i, x_i; w, b) + \lambda \|w\|_2^2$$

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard SVM solution.
Set hyperparameters $\mu$ and $\lambda$.
**repeat**
Compute $s_{wi} \in \frac{\partial h_\mu(w^k,b^k)}{\partial w}$ and $s_{bi} \in \frac{\partial h_\mu(w^k,b^k)}{\partial b}$.
Solve the quadratic program to obtain $w^{k+1}, b^{k+1}$,

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{\mu}\sum\limits_{i=1}^{n} \xi_i + \lambda \langle w, w\rangle - \langle s_w, w\rangle - s_b b \\ \text{subject to} & \xi_i \geqslant \mu - y_i(\langle w, x_i\rangle + b) \\ & \xi_i \geqslant 0. \end{array}$$

**until** $F_{A4}(w^{k+1}, b^{k+1}) = F_{A4}(w^k, b^k)$

### A.5. $\ell_1$ *with ramp loss*

This was recently proposed in [7]. The objective function is given by

$$F_{A5}(w,b) = \sum_{i=1}^{n} \ell_\mu(y_i, x_i; w, b) + \lambda \sum_{j=1}^{d} |w_j|$$

**Initialization:** Set $k = 0$ and initialize $w^0, b^0$ with a standard 1-norm SVM solution.
Set hyperparameters $\mu$ and $\lambda$.
**repeat**
Compute $s_{wi} \in \frac{\partial h_\mu(w^k,b^k)}{\partial w}$, and $s_{bi} \in \frac{\partial h_\mu(w^k,b^k)}{\partial b}$.
Solve the linear program to obtain $w^{k+1}, b^{k+1}$,

$$\begin{array}{ll} \min\limits_{w,b,\xi,\eta} & \frac{1}{\mu}\sum\limits_{i=1}^{n} \xi_i + \lambda \sum\limits_{j=1}^{d} \eta_j - \langle s_w, w\rangle - s_b b \\ \text{subject to} & \xi_i \geqslant \mu - y_i(\langle w, x_i\rangle + b) \\ & -\eta_j \leqslant w_j \leqslant \eta_j \\ & \xi_i, \eta_j \geqslant 0. \end{array}$$

**until** $F_{A5}(w^{k+1}, b^{k+1}) = F_{A5}(w^k, b^k)$

### A.6. *Details of UCI data*

In Table A1, we show some details of the datasets obtained from `http://archive.ics.uci.edu/ml/datasets.html`.

| Name | examples | features | Full name |
|---|---|---|---|
| pima | 768 | 8 | Pima Indians Diabetes |
| bupa | 345 | 7 | Liver Disorders |
| ionosphere | 351 | 34 | Ionosphere |
| wpbc | 198 | 34 | Breast Cancer Wisconsin (Prognostic) |
| sonar | 208 | 60 | Connectionist Bench (Sonar, Mines vs. Rocks) |

Table A1. Details of UCI data

### A.7. *Details of feature selection challenge data*

In Table A2, we show some details of the datasets obtained from `http://www.nipsfsc.ecs.soton.ac.uk/datasets/`, where we merged the training and validation sets.

| Name | examples | features |
|---|---|---|
| Arcene | 10000 | 200 |
| Gisette | 5000 | 7000 |

Table A2. Details of the feature selection challenge data

## References

[1] Asa Ben-Hur, Cheng Soon Ong, Sören Sonnenburg, Bernhard Schölkopf, and Gunnar Rätsch. Support vector machines and kernels for computational biology. *PLoS Computational Biology*, 4(10):e1000173, 2008.

[2] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

[3] Phil Long and Rocco Servedio. Random classification noise defeats all convex potential boosters. In *International Conference on Machine Learning*, 2008.

[4] Yoav Freund. A more robust boosting algorithm. Technical Report arXiv:0905.2138.v1, Computer Science and Engineering, UCSD, 2009.

[5] Ronan Collobert, Fabian Sinz, Jason Weston, and Leon Bottou. Trading convexity for scalability. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 275–300, 2007.

[6] Chuong B. Do, Quoc Le, Choon Hui Teo, Olicier Chapelle, and Alex Smola. Tighter bounds for structured estimation. In *Neural Information Processing Systems*, 2008.

[7] Yichao Wu and Yufeng Liu. Robust truncated-hinge-loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.

[8] I. Steinwart, D. Hush, and C. Scovel. An oracle inequality for clipped regularized risk minimizers. In *Neural Information Processing Systems*, pages 1321–1328, 2007.

[9] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Information Science and Statistics. Springer Verlag, 2008.

[10] Hoai An Le Thi, Hoai Minh Le, Van Vinh Nguyen, and Tao Pham Dinh. A DC programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, 2(3):259–278, 2008.

[11] Dori Peleg and Ron Meir. A bilinear formulation for vector sparsity optimization. *Signal Processing*, 88:375–389, 2008.

[12] Tong Zhang. Some sharp performance bounds for least squares regression with l1 regularization. *Annals of Statistics*, 37(5A):2109–2144, 2009.

[13] Tao Pham Dinh and Hoai An Le Thi. DC optimization algorithms for solving the trust region subproblem. *SIAM Journal on Optimization*, 8(2):476–505, 1998.

[14] Hoai An Le Thi and Tao Pham Dinh. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[15] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems*, 2004.

[16] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[17] Jun Zhu and Eric P. Xing. On primal and dual sparsity of markov networks. In *International Conference on Machine Learning*, 2009.

[18] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[19] Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1):246–270, 2009.

[20] Olvi L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

[21] Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of non-convex penalties and DC programming. *IEEE Transactions on Signal Processing*, 57(12):4686–4698, 2009.

[22] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neur. Comp.*, 15(4):915–936, 2003.

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

[24] N. Krause and Y. Singer. Leveraging the margin more carefully. In *Proceedings of the twenty-first international conference on Machine learning*, 2004.

[25] Y. Liu, X. Shen, and H. Doss. Multicategory $\psi$-learning and support vector machine: Computational tools. *Journal of Computational and Graphical Statistics*, 14:219–236, 2005.

[26] Y. Liu and X. Shen. Multicategory $\psi$-learning. *Journal of the American Statistical Association*, 101:500–509, 2006.

[27] Hoai An Le Thi, T. Belghiti, and Tao Pham Dinh. A new efficient algorithm based on DC programming and DCA for clustering. *Journal of Global Optimization*, 37:593–608, 2006.

[28] Hoai An Le Thi, Hoai Minh Le, and Tao Pham Dinh. Optimization based DC programming and DCA for hierarchical clustering. *European Journal of Operational Research*, 183(1067–1085), 2007.

[29] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of International Conference on Machina Learning*, 1998.

[30] Tao Pham Dinh. Algorithms for solving a class of non convex optimization problems. methods of subgradients. In *Fermat Days 85. Mathematics for Optimization*. Elsevier Science Publishers, B.V. North-Holland., 1986.

[31] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer Verlag, 2001.

[32] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[33] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.