Version of Record: https://www.sciencedirect.com/science/article/pii/S0893608020303233
Manuscript_845272159d3c4fd2e95a27c04131e8d2

Stochastic DCA for minimizing a large sum of DC functions with application to Multi-class Logistic Regression

Hoai An Le Thi[a,b,c,*], Hoai Minh Le[c], Duy Nhat Phan[c], Bach Tran[c]

[a]*Department for Management of Science and Technology Development, Ton Duc Thang University, Ho Chi Minh City, Vietnam*

[b]*Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam*

[c]*Department of Computer Science and Applications, LGIPM, University of Lorraine, France*

Abstract

We consider the large sum of DC (Difference of Convex) functions minimization problem which appear in several different areas, especially in stochastic optimization and machine learning. Two DCA (DC Algorithm) based algorithms are proposed: stochastic DCA and inexact stochastic DCA. We prove that the convergence of both algorithms to a critical point is guaranteed with probability one. Furthermore, we develop our stochastic DCA for solving an important problem in multi-task learning, namely group variables selection in multi class logistic regression. The corresponding stochastic DCA is very inexpensive, all computations are explicit. Numerical experiments on several benchmark datasets and synthetic datasets illustrate the efficiency of our algorithms and their superiority over existing methods, with respect to classification accuracy, sparsity of solution as well as running time.

Keywords: Large sum of DC functions, DC Programming, DCA, Stochastic DCA, Inexact Stochastic DCA, Multi-class Logistic Regression

*Corresponding author

Email addresses: lethihoaiant@tdtu.edu.vn (Hoai An Le Thi), hoai-an.le-thi@univ-lorraine.fr (Hoai An Le Thi), minh.le@univ-lorraine.fr (Hoai Minh Le), duy-nhat.phan@univ-lorraine.fr (Duy Nhat Phan), bach.tran@univ-lorraine.fr (Bach Tran)

© 2020 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

1. Introduction

We address the so-called *large sum of DC functions minimization* problem which takes the form

$$\min_{x\in\mathbb{R}^d}\left\{F(x) := \frac{1}{n}\sum_{i=1}^{n} F_i(x)\right\}, \tag{1}$$

where F_i are DC functions, i.e., $F_i(x) = g_i(x) - h_i(x)$ with g_i being lower semi-continuous proper convex and h_i being convex, and n is a very large integer number. The problem of minimizing F over a convex set Ω is also of the type (1), as the convex constraint $x \in \Omega$ can be incorporated into the objective function F via the indicator function χ_Ω of Ω defined by $\chi_\Omega(x) = 0$ if $x \in \Omega$, $+\infty$ otherwise. Our study is motivated by the fact that the problem (1) appears in several different contexts, especially in stochastic optimization and machine learning. For instance, let us consider the minimization of expected loss in stochastic programming

$$\min_{x\in\Omega} \mathbb{E}[f(x,\xi)], \tag{2}$$

where f is a loss function of variables x and ξ, and ξ is a random variable. A standard approach for solving (3) is the sample average method (Healy & Schruben, 1991) which approximates the problem (2) by

$$\min_{x\in\Omega} \frac{1}{n}\sum_{i=1}^{n} f(x,\xi_i), \tag{3}$$

where $\xi_1, ..., \xi_n$ are independent, identically distributed realizations of ξ. When the loss function f is DC, the problem (3) takes the form of (1) with $F_i(x) = f(x,\xi_i) + \chi_\Omega(x)$. Obviously, the larger n is, the better approximation will be. Hence, a good approximate model of the form (3) in average sample methods requires an extremely large number n.

Furthermore, let us consider an important problem in machine learning, the multi-task learning. Let T be the number of tasks. For the j-th task, the training set $\mathcal{D}_j$ consists of n_j labeled data points in the form of ordered pairs (x_i^j, y_i^j), $i = 1, ..., n_j$, with $x_i^j \in \mathbb{R}^d$ and its corresponding output $y_i^j \in \mathbb{R}$. Multi-task learning aims to estimate T predictive functions $f_\theta^j(x) : \mathbb{R}^d \to \mathbb{R}, j = 1, ..., T$, which fit well the data.

The multi-task learning can be formulated as

$$\min_{\theta} \left\{ \sum_{j=1}^{T} \sum_{i=1}^{n_j} \mathcal{L}(y_i^j, f_\theta^j(x_i^j)) + \lambda p(\theta) \right\}, \tag{4}$$

where $\mathcal{L}$ denotes the loss function, p is a regularization term and $\lambda > 0$ is a trade-off hyper-parameter. For a good learning process, $\sum_{j=1}^{T} n_j$ is, in general, a very large number. Clearly, this problem takes the form of (1) when $\mathcal{L}$ and p are DC functions. We observe that numerous loss functions in machine learning (e.g. least square loss, squared hing loss, ramp loss, logistic loss, sigmoidal loss, etc) are DC. On another hand, most of the existing regularizations can be expressed as DC functions. For instance, in learning with sparsity problems involving the zero norm (which include, among of others, variable / group variable selection in classification, sparse regression, compressed sensing) all standard nonconvex regularizations studied in the literature are DC functions (Le Thi et al., 2015). Moreover, in many applications dealing with big data, the number of both variables and samples are very large.

The problem (1) has a double difficulties due to the nonconvexity of F_i and the large value of n. Meanwhile, the sum structure of F enjoys an advantage: one can work on F_i instead of the whole function F. Since all F_i are DC functions, F is DC too, and therefore (1) is a standard DC program, i.e., minimizing a DC function under a convex set and/or the whole space.

To the best of our knowledge, although several methods have been developed for solving different special cases of (1), there is no existing work that considers the general problem (1) as well. The stochastic gradient (SG) method was first introduced in Robbins & Monro (1951) and then developed in Bottou (1998); LeCun et al. (1998) for solving (3) in the unconstrained case ($\Omega = R^d$) with $f(\cdot, \xi_i)$ being smooth functions. The SG method chooses $i_l \in \{1, ..., n\}$ randomly and performs the update

$$x^{l+1} = x^l - \alpha_l \nabla f(x^l, \xi_{i_l}), \tag{5}$$

where α_l is the step size and $\nabla f(x^l, \xi_{i_l})$ is a stochastic gradient. Later, Bertsekas (2011, 2010) proposed the proximal stochastic subgradient methods (also referred as incremental proximal methods) for solving (3) in convex case, i.e., Ω is a closed convex set and $f(\cdot, \xi_i)$ are convex functions. The computational cost per iteration of these

basic SG methods is very cheap, however, due to the variance introduced by random sampling, their convergence rates are slower than the "full" gradient methods. Hence, some SG methods for solving (3) in unconstrained differentiable convex case use either the average of the stored past gradients or a multi-stage scheme to progressively reduce the variance of the stochastic gradient (see e.g Schmidt et al. (2017); Shalev-Schwartz & Zhang (2013); Defazio et al. (2014a,b); Johnson & Zhang (2013)). With the variance reduction techniques, other variants of the SG method have been proposed for nonconvex problem (3) where the L-smooth property is required (see e.g. Mairal (2015); Reddi et al. (2016); Allen-Zhu & Yuan (2016)).

As (1) is a DC program, a natural way to tackle it is using DCA (DC Algorithm) (see (Le Thi & Pham Dinh, 2005, 2018; Pham Dinh & Le Thi, 1998, 1997, 2014) and references therein), an efficient approach in nonconvex programming framework. DCA addresses the problem of minimizing a DC function on the whole space $\mathbb{R}^d$ or on a closed convex set $\Omega \subset \mathbb{R}^d$. Generally speaking, a standard DC program takes the form:

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

where G, H are lower semi-continuous proper convex functions on $\mathbb{R}^d$. Such a function F is called a DC function, and $G - H$ is a DC decomposition of F while G and H are the DC components of F. DCA has been introduced in 1985 Pham Dinh & Souad (1986) and extensively developed since 1993 ((Le Thi & Pham Dinh, 2005, 2018; Pham Dinh & Le Thi, 1998, 1997, 2014) and references therein) to become now classic and increasingly popular. Most existing methods in convex/nonconvex programming are special versions of DCA via appropriate DC decompositions (see (Le Thi & Pham Dinh, 2018)). In recent years, numerous DCA-based algorithms have been developed for successfully solving large-scale nonsmooth/nonconvex programs appearing in several application areas, especially in machine learning, communication system, biology, finance, etc. (see e.g. the list of references in Le Thi (Home Page); Le Thi & Pham Dinh (2018)). DCA has been proved to be a fast and scalable approach which is, thanks to the effect of DC decompositions, more efficient than related methods. For a comprehensible survey on thirty years of development of DCA, the

reader is referred to the recent paper (Le Thi & Pham Dinh, 2018). New trends in the development of DCA concern novel versions of DCA-based algorithms (e.g. on-line/stochastic/approximate/like DCA) to accelerate the convergence and to deal with large-scale setting and big data. Our present work follows this direction.

The original key idea of DCA relies on the DC structure of the objective function F. DCA consists in iteratively approximating the considered DC program by a sequence of convex ones. More precisely, at each iteration l, DCA approximates the second DC component $H(x)$ by its affine minorization $H_l(x) := H(x^l) + \langle x - x^l, y^l \rangle$, with $y^l \in \partial H(x^l)$, and minimizes the resulting convex function.

Basic DCA scheme

Initialization: Let $x^0 \in \text{dom}\, \partial H$, $l = 0$.

For $l = 0, 1, \ldots$ **until convergence of** $\{x^l\}$**:**

k1: Calculate $y^l \in \partial H(x^l)$;

k2: Calculate $x^{l+1} \in \text{argmin}\{G(x) - H_l(x) : x \in \mathbb{R}^d\}$ (P_l).

Paper's contributions. We first propose the so-called *stochastic DCA*, by taking the advantage of the sum structure of F, for solving (1). For the first time in the literature a stochastic algorithm is investigated to the general problem (1) which is, as mentioned above, a common model of various applications. Exploiting the efficiency of DCA, we develop novel DCA-based algorithms to tackle the difficulty due to the large value of n. The basic idea of stochastic DCA is to update, at each iteration, the minorant of only some randomly chosen h_i while keeping the minorant of the other h_i. Hence the main advantage of the stochastic DCA versus standard DCA is the computational reduction in the step of computing a subgradient of H. Meanwhile, the convex subproblem is the same in both standard DCA and stochastic DCA. The first work in this direction was published in the conference paper Le Thi et al. (2017) where we only considered a machine learning problem which is a special case of (1), namely

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \lambda \|x\|_{2,0},$$

where f_i are L-Lipschitz functions. We rigorously studied the convergence properties of this stochastic DCA and proved that its convergence is guaranteed with probability

one. In the present work, we also prove the same convergence properties of stochastic DCA for the general problem (1) where F_i is any DC function.

Secondly, to deal with the computational aspect in large-scale setting, we further propose an *inexact stochastic DCA* version in which both subgradient of H and optimal solution of the resulting convex program are approximately computed. We show that the convergence properties of stochastic DCA are still valid for the inexact stochastic DCA.

Thirdly, we show how to develop the proposed stochastic DCA for the group variables selection in multi-class logistic regression, a very important problem in machine learning which takes the form (1). We consider three nonconvex regularizations $\ell_{q,0}$ ($q \in \{1, 2, \infty\}$) with two approximation functions of $\ell_{q,0}$, and develop 12 algorithms (6 versions of standard DCA and 6 versions of stochastic DCA) for this problem. For the first time, a careful study is proposed in the literature. Thanks to a suitable DC decomposition, all versions of our proposed algorithms are explicit, i.e. all calculations are explicitly defined and no supplement solver is needed. Numerical experiments on very large synthetic and real-world datasets show that our approach is much more efficient, in both quality and rapidity, than four related methods.

The remainder of the paper is organized as follows. Solution methods based on stochastic DCA for solving (1) are developed in Section 2 while the stochastic DCA for the group variables selection in multi-class logistic regression is presented in Section 3. Numerical experiments are reported in Section 4. Finally, Section 5 concludes the paper.

2. Stochastic DCA for minimizing a large sum of DC functions

Before presenting the stochastic DCA, let us recall some basic notations that will be used in the sequel.

The modulus of a convex function $\theta : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ on Ω, denoted by $\rho(\theta, \Omega)$ or $\rho(\theta)$ if $\Omega = \mathbb{R}^n$, is given by

$$\rho(\theta, \Omega) = \sup\{\rho \geq 0 : \theta - (\rho/2)\|.\|^2 \text{ is convex on } \Omega\}.$$

One says that θ is ρ-convex (resp. *strongly convex*) on Ω if $\rho(\theta, \Omega) \geq 0$ (resp. $\rho(\theta, \Omega) > 0$).

For $\varepsilon > 0$ and $x^0 \in \text{dom}\,\theta$, the ε-subdifferential of θ at x^0, denoted $\partial\theta_\varepsilon(x^0)$, is defined by

$$\partial\theta_\varepsilon(x^0) := \{y \in \mathbb{R}^d : \theta(x) \geq \theta(x^0) + \langle x - x^0, y\rangle - \varepsilon,\ \forall x \in \mathbb{R}^d\}, \tag{6}$$

while $\partial\theta(x^0)$ stands for the usual (or exact) subdifferential of θ at x^0 (i.e. $\varepsilon = 0$ in (6)).

For $\epsilon \geq 0$, a point x_ϵ is called an ϵ-solution of the problem $\inf\{f(x) : x \in \mathbb{R}^d\}$ if

$$f(x_\epsilon) \leq f(x) + \epsilon\ \forall x \in \mathbb{R}^d.$$

2.1. Stochastic DCA

Now, let us introduce a stochastic version of DCA, named SDCA, for solving (1). A natural DC formulation of the problem (1) is

$$\min\left\{F(x) = G(x) - H(x) : x \in \mathbb{R}^d\right\}, \tag{7}$$

where

$$G(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x) \text{ and } H(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x).$$

According to the generic DCA scheme, DCA for solving the problem (7) consists of computing, at each iteration l, a subgradient $v^l \in \partial H(x^l)$ and solving the convex subproblem of the form

$$\min\left\{G(x) - \langle v^l, x\rangle : x \in \mathbb{R}^d\right\}. \tag{8}$$

As $H = \frac{1}{n}\sum_{i=1}^{n} h_i$, the computation of subgradients of H requires the one of all functions h_i. This may be expensive when n is very large. The main idea of SDCA is to update, at each iteration, the computation of subgradient of only some randomly chosen h_i while keeping the one of the other h_i.

SDCA for solving the problem (7) is described in Algorithm 1 below.

Algorithm 1 SDCA for solving the problem (1)

Initialization: Choose $x^0 \in \text{dom}\, \partial H$, $s_0 = \{1, ..., n\}$, and $l = 0$.

Repeat

1. Compute $v_i^l \in \partial h_i(x^l)$ if $i \in s_l$ and keep $v_i^l = v_i^{l-1}$ if $i \notin s_l$, $l > 0$. Set $v^l = \frac{1}{n}\sum_{i=1}^n v_i^l$.

2. Compute x^{l+1} by solving the convex problem (8).

3. Set $l = l + 1$ and randomly choose a small subset $s_l \subset \{1, ..., n\}$.

Until Stopping criterion.

Remark 1. *SDCA differs from DCA only at the step* 1. *Instead of computing* ∂H *(by computing* ∂h_i *for all* $i = 1, \ldots, n$*) as in DCA, SDCA updates only a small number of* ∂h_i*. Clearly, this technique reduces considerably the running time of DCA, and, intuitively, SDCA should be much faster than DCA, in particular when the computation of* ∂h_i *is expensive. Naturally, such a technique may reduce the quality of solution, and the crucial matter is the convergence properties of SDCA: do we get a critical point as in DCA ?*

The following theorem shows that the convergence properties of SDCA are guaranteed with probability one.

Theorem 1. *Assume that* $\alpha^* = \inf F(x) > -\infty$*, and* $|s_l| = b$ *for all* $l > 0$*. Let* $\{x^l\}$ *be a sequence generated by SDCA , the following statements hold.*

a) $\{F(x^l)\}$ *is the almost sure convergent sequence.*

b) If $\min_i \rho(h_i) > 0$*, then* $\sum_{l=1}^{\infty} \|x^l - x^{l-1}\|^2 < +\infty$ *and* $\lim_{l\to\infty} \|x^l - x^{l-1}\| = 0$*, almost surely.*

c) If $\min_i \rho(h_i) > 0$*, then every limit point of* $\{x^l\}$ *is a critical point of* F *with probability one.*

Proof. a) Let x_i^0 be the copies of x^0. We set $x_i^{l+1} = x^{l+1}$ for all $i \in s_{l+1}$ and $x_j^{l+1} = x_j^l$ for $j \notin s_{l+1}$. We then have $v_i^l \in \partial h_i(x_i^l)$ for $i = 1, ..., n$. Let T_i^l be

the function given by

$$T_i^l(x) = g_i(x) - h_i(x_i^l) - \left\langle x - x_i^l, v_i^l \right\rangle.$$

It follows from $v_i^l \in \partial h_i(x_i^l)$ that

$$h_i(x) \geq h_i(x_i^l) + \left\langle x - x_i^l, v_i^l \right\rangle.$$

That implies $T_i^l(x) \geq F_i(x) \geq F_i(x)$ for all $l \geq 0$, $i = 1, ..., n$. We also observe that x^{l+1} is a solution to the following convex problem

$$\min_x T^l(x) := \frac{1}{n} \sum_{i=1}^n T_i^l(x), \tag{9}$$

Therefore

$$\begin{aligned} T^l(x^{l+1}) \leq T^l(x^l) &= T^{l-1}(x^l) + \frac{1}{n} \sum_{i \in s_l} [T_i^l(x^l) - T_i^{l-1}(x^l)] \\ &= T^{l-1}(x^l) + \frac{1}{n} \sum_{i \in s_l} [F_i(x^l) + 2\epsilon^l - T_i^{l-1}(x^l)], \end{aligned} \tag{10}$$

where the second equality follows from $T_i^l(x^l) = F_i(x^l)$ for all $i \in s_l$. Let $\mathcal{F}_l$ denote the σ-algebra generated by the entire history of SDCA up to the iteration l, i.e., $\mathcal{F}_0 = \sigma(x^0)$ and $\mathcal{F}_l = \sigma(x^0, ..., x^l, s_0, ..., s_{l-1})$ for all $l \geq 1$. By taking the expectation of the inequality (A.2) conditioned on $\mathcal{F}_l$, we have

$$\mathbb{E}\left[T^l(x^{l+1}) | \mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{n}\left[T^{l-1}(x^l) - F(x^l)\right].$$

By applying the supermartingale convergence theorem (Neveu, 1975; Bertsekas et al., 2003) to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}$, $\{\frac{b}{n}[T^{l-1}(x^l) - F(x^l)]\}$ and $\{0\}$, we conclude that the sequence $\{T^{l-1}(x^l, y^l) - \alpha^*\}$ converges to $T^* - \alpha^*$ and

$$\sum_{l=1}^{\infty} \left[T^{l-1}(x^l) - F(x^l)\right] < \infty, \tag{11}$$

with probability 1. Therefore $\{F(x^l)\}$ converges almost surely to T^*.

b) By $v_i^{l-1} \in \partial h_i(x_i^{l-1})$, we have

$$h_i(x) \geq h_i(x_i^{l-1}) + \langle x - x_i^{l-1}, v_i^{l-1} \rangle + \frac{\rho(h_i)}{2} \|x - x_i^{l-1}\|^2, \ \forall x \in \mathbb{R}^d.$$

This implies

$$F_i(x) \leq T_i^{l-1}(x) - \frac{\rho(h_i)}{2}\|x - x_i^{l-1}\|^2. \tag{12}$$

From (A.2) and (A.4) with $x = x^l$, we have

$$T^l(x^{l+1}) \leq T^{l-1}(x^l) - \frac{1}{n}\sum_{i\in s_l}\frac{\rho(h_i)}{2}\|x - x_i^{l-1}\|^2. \tag{13}$$

Taking the expectation of the inequality (A.5) conditioned on $\mathcal{F}_l$, we obtain

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{4n^2}\sum_{i=1}^{n}\rho(h_i)\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Combining this and $\rho = \min_{i=1,\ldots,n}\rho(h_i) > 0$ gives us

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b\rho}{2n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2.$$

Applying the supermartingale convergence theorem to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}$, $\{\frac{b\rho}{2n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2\}$ and $\{0\}$, we get

$$\sum_{l=1}^{\infty}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 < \infty,$$

with probability 1. In particular, for $i = 1, ..., n$, we have

$$\sum_{l=1}^{\infty}\|x^l - x_i^{l-1}\|^2 < \infty, \tag{14}$$

and hence $\lim_{l\to\infty}\|x^l - x_i^{l-1}\| = 0$ almost surely.

c) Assume that there exists a sub-sequence $\{x^{l_k}\}$ of $\{x^l\}$ such that $x^{l_k} \to x^*$ almost surely. From (A.6), we have $\|x^{l_k+1} - x_i^{l_k}\| \to 0$ almost surely. Therefore, by the finite convexity of h_i, without loss of generality, we can suppose that the sub-sequence $v_i^{l_k}$ tends to v_i^* almost surely. Since $v_i^{l_k} \in \partial h_i(x_i^{l_k})$ and by the closed property of the subdifferential mapping ∂h_i, we have $v_i^* \in \partial h_i(x^*)$. As x^{l_k+1} is a solution of the problem $\min_x T^{l_k}(x)$, we obtain

$$0 \in \partial T^{l_k}(x^{l_k+1}). \tag{15}$$

This is equivalent to

$$0 \in \partial\frac{1}{n}\sum_{i=1}^{n}g_i(x^{l_k+1}) - \frac{1}{n}\sum_{i=1}^{n}v_i^{l_k} = \partial G(x^{l_k+1}) - \frac{1}{n}\sum_{i=1}^{n}v_i^{l_k}. \tag{16}$$

Hence, $\frac{1}{n}\sum_{i=1}^{n} v_i^{l_k} \in \partial G(x^{l_k+1})$. By the closedness property of the subdifferential mapping ∂G, we obtain $v^* = \frac{1}{n}\sum_{i=1}^{n} v_i^* \in \partial G(x^*)$ with probability one. Therefore,

$$v^* \in \partial G(x^*) \cap \partial H(x^*), \tag{17}$$

with probability 1. This implies that x^* is a critical point of F with probability 1 and the proof is then complete. □

2.2. Inexact stochastic DCA

The SDCA scheme requires the exact computations of v_i^l and x^{l+1}. Observing that, for standard DCA these computations are not necessarily exact (Le Thi & Pham Dinh (2018)), we are suggested to introduce an inexact version of SDCA. This could be useful when the exact computations of v_i^l and x^{l+1} are expensive. The inexact version of SDCA computes ϵ-subgradients $v_i^l \in \partial_{\epsilon^l} h_i(x^l)$ and an ϵ^l-solution x^{l+1} of the convex problem (8) instead of exactly computing. The inexact version of SDCA, named ISDCA, is described as follows.

Algorithm 2 ISDCA for solving the problem (1)

Initialization: Choose $x^0 \in \text{dom}\, \partial H$, $s_0 = \{1, ..., n\}$, $\epsilon^0 \geq 0$ and $l = 0$.

Repeat

1. Compute $v_i^l \in \partial_{\epsilon^l} h_i(x^l)$ if $i \in s_l$ and keep $v_i^l = v_i^{l-1}$ if $i \notin s_l$, $l > 0$. Set $v^l = \frac{1}{n}\sum_{i=1}^{n} v_i^l$.

2. Compute an ϵ^l-solution x^{l+1} of the convex problem (8).

3. Set $l = l + 1$, randomly choose a small subset $s_l \subset \{1, ..., n\}$, and update $\epsilon^l \geq 0$.

Until Stopping criterion.

Remark 2. *ISDCA should be less expensive than SDCA when the computation of ∂h_i and/or the solution of convex subproblems* (8) *is very hard and expensive. In such cases ISDCA could be an effective algorithm. If the computation of all ∂h_i as well as the solution of convex subproblems* (8) *are defined in explicit forms, then this version ISDCA is useless.*

Under an assumption that $\sum_{l=0}^{\infty} \epsilon^l < +\infty$, the ISDCA has the same convergence properties as SDCA, which are stated in the following theorem.

Theorem 2. *Assume that $\alpha^* = \inf F(x) > -\infty$, and $|s_l| = b$ for all $l > 0$. Let $\{x^l\}$ be a sequence generated by ISDCA with respect to a nonnegative sequence $\{\epsilon^l\}$ such that $\sum_{l=0}^{\infty} \epsilon^l < +\infty$ almost surely. The following statements hold.*

a) $\{F(x^l)\}$ is the almost sure convergent sequence.

b) If $\min_i \rho(h_i) > 0$, then $\sum_{l=1}^{\infty} \|x^l - x^{l-1}\|^2 < +\infty$ and $\lim_{l\to\infty} \|x^l - x^{l-1}\| = 0$, almost surely.

c) If $\min_i \rho(h_i) > 0$, then every limit point of $\{x^l\}$ is a critical point of F with probability one.

This theorem is analogously proved as Theorem 1 and its proof is provided in Appendix A.

3. Application to Group Variables Selection in multi-class Logistic Regression

Logistic regression, introduced by D. Cox in 1958 Cox (1958), is undoubtedly one of the most popular supervised learning methods. Logistic regression has been successfully applied in various real-life problems such as cancer detection Kim et al. (2008), medical Boyd et al. (1987); Bagley et al. (2001); Subasi & Erçelebi (2005), social science King & Zeng (2001), etc. Especially, logistic regression combined with feature selection has been proved to be suitable for high dimensional problems, for instance, document classification Genkin et al. (2007) and microarray classification Liao & Chin (2007); Kim et al. (2008).

The multi-class logistic regression problem can be described as follows. Let $\{(x_i, y_i) : i = 1, ..., n\}$ be a training set with observation vectors $x_i \in \mathbb{R}^d$ and labels $y_i \in \{1, ..., Q\}$ where Q is the number of classes. Let W be the $d \times Q$ matrix whose columns are $W_{:,1}, ..., W_{:,Q}$ and $b = (b_1, ..., b_Q) \in \mathbb{R}^Q$. The couple $(W_{:,i}, b_i)$ forms the hyperplane $f_i := W_{:,i}^T x + b_i$ that separates the class i from the other classes.

In the multi-class logistic regression problem, the conditional probability $p(Y = y|X = x)$ that an instance x belongs to a class y is defined as

$$p(Y = y|X = x) = \frac{\exp(b_y + W_{:,y}^T x)}{\sum\limits_{k=1}^{Q} \exp(b_k + W_{:,k}^T x)}. \tag{18}$$

We aim to find (W, b) for which the total probability of the training observations x_i belonging to its correct classes y_i is maximized. A natural way to estimate (W, b) is to minimize the negative log-likelihood function which is defined by

$$\mathcal{L}(W, b) := \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) \tag{19}$$

where $\ell(x_i, y_i, W, b) = -\log p(Y = y_i|X = x_i)$. Moreover, in high-dimensional settings, there are many irrelevant and/or redundant features. Hence, we need to select important features in order to reduce overfitting of the training data. A feature j is to be removed if and only if all components in the row j of W are zero. Therefore, it is reasonable to consider rows of W as groups. Denote by $W_{j,:}$ the j-th row of the matrix W. The $\ell_{q,0}$-norm of W, i.e., the number of non-zero rows of W, is defined by

$$\|W\|_{q,0} = |\{j \in \{1, ..., d\} : \|W_{j,:}\|_q \neq 0\}|.$$

Hence, the $\ell_{q,0}$ regularized multi-class logistic regression problem is formulated as follows

$$\min_{W,b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) + \lambda \|W\|_{q,0} \right\}. \tag{20}$$

In this application, we use a non-convex approximation of the $\ell_{q,0}$-norm based on the following two penalty functions $\eta_\alpha(s)$:

$$\begin{aligned} \text{Exponential:} \quad & \eta_\alpha^{\text{exp}}(s) \quad = 1 - \exp(-\alpha s), \\ \text{Capped-}\ell_1\text{:} \quad & \eta_\alpha^{\text{cap-}\ell_1}(s) \quad = \min\{1, \alpha s\}. \end{aligned}$$

These penalty functions have shown their effectiveness in several problems, for instance, individual variables selection in SVM Bradley & Mangasarian (1998); Le Thi et al. (2008), sparse optimal scoring problem Le Thi & Phan (2016), sparse covariance matrix estimation problem Phan et al. (2017), and bi-level/group variables selection

Le Thi et al. (2019); Phan & Thi (2019). For a more complete references on nonconvex approximation approaches for the ℓ_0 norm, refer to Le Thi et al. (2015).

The corresponding approximate problem of (20) takes the form:

$$\min_{W,b}\left\{\frac{1}{n}\sum_{i=1}^{n}\ell(x_i,y_i,W,b)+\lambda\sum_{j=1}^{d}\eta_\alpha(\|W_{j,:}\|_q)\right\}. \tag{21}$$

Since η_α is increasing on $[0,+\infty)$, the problem (21) can be equivalently reformulated as follows

$$\min_{(W,b,t)}\left\{\frac{1}{n}\sum_{i=1}^{n}\left[\ell(x_i,y_i,W,b)+\chi_\Omega(W,b,t)+\lambda\sum_{j=1}^{d}\eta_\alpha(t_j)\right]\right\}, \tag{22}$$

where $\Omega=\{(W,b,t)\in\mathbb{R}^{d\times Q}\times\mathbb{R}^Q\times\mathbb{R}^d:\|W_{j,:}\|_q\le t_j, j=1,...,d\}$. Moreover, as $\ell(x_i,y_i,W,b)$ is differentiable with L-Lipschitz continuous gradient and η_α is concave, the problem (22) takes the form of (1) where the function $F_i(W,b,t)$ is given by

$$F_i(W,b,t)=\ell(x_i,y_i,W,b)+\chi_\Omega(W,b,t)+\lambda\sum_{j=1}^{d}\eta_\alpha(t_j):=g_i(W,b,t)-h_i(W,b,t),$$

where the DC components g_i and h_i are defined by (with $\rho>L$):

$$g_i(W,b,t)=\frac{\rho}{2}\|(W,b)\|^2+\chi_\Omega(W,b,t),$$
$$h_i(W,b,t)=\frac{\rho}{2}\|(W,b)\|^2-\ell(x_i,y_i,W,b)-\lambda\sum_{j=1}^{d}\eta_\alpha(t_j).$$

Before presenting SDCA for solving (22), let us show how to apply standard DCA to this problem.

3.1. Standard DCA for solving the problem (22)

We consider three norms corresponding to $q\in\{1,2,\infty\}$. DCA applied to (22) consists of computing, at each iteration l, $(U^l,v^l,z^l)\in\partial H(W^l,b^l,t^l)$, and solving the convex sub-problem

$$\min_{(W,b,t)}\left\{\frac{\rho}{2}\|(W,b)\|^2+\chi_\Omega(W,b,t)-\langle U^l,W\rangle-\langle v^l,b\rangle-\langle z^l,t\rangle\right\}. \tag{23}$$

The computation of (U^l,v^l,z^l) is explicitly defined as follows.

$$(U^l,v^l,z^l)=\frac{1}{n}\sum_{i=1}^{n}(U_i^l,v_i^l,z_i^l),(U_i^l,v_i^l,z_i^l)\in\partial h_i(W^l,b^l,t^l),$$

$$
\begin{aligned}
(U_i^l)_{:,k} &= \rho W_{:,k}^l - \left(p_k^l(x_i) - \delta_{ky_i}\right) x_i, k = 1, ...Q, \\
(v_i^l)_k &= \rho b_k^l - \left(p_k^l(x_i) - \delta_{ky_i}\right), k = 1, ...Q, \\
(z_i^l)_j &= \begin{cases} -\lambda\alpha \exp(-\alpha t_j^l), & j = 1, \ldots, d \quad \text{if } \eta_\alpha = \eta_\alpha^{\exp}, \\ -\lambda\alpha \text{ if } \alpha t_j^l \leq 1, \text{ and } 0 \text{ otherwise}, & j = 1, \ldots, d, \quad \text{if } \eta_\alpha = \eta_\alpha^{\text{cap}-\ell_1}, \end{cases}
\end{aligned} \tag{24}
$$

with $p_k^l(x_i) = \exp(b_k^l + (W_{:,k}^l)^T x_i)/(\sum_{h=1}^Q b_h^l + (W_{:,h}^l)^T x_i))$, $\delta_{ky_i} = 1$ if $k = y_i$ and 0 otherwise.

The convex sub-problem (23) can be solved as follows (note that $z_j^l \leq 0$ for $j = 1, \ldots, d$)

$$
W^{l+1} = \arg\min_W \left\{ \frac{\rho}{2}\|W\|^2 + \sum_{j=1}^d (-z_j^l)\|W_{j,:}\|_q - \langle U^l, W\rangle \right\}, \tag{25}
$$

$$
b^{l+1} = \arg\min_b \left\{ \frac{\rho}{2}\|b\|^2 - \langle v^l, b\rangle \right\} = \frac{1}{\rho} v^l, \tag{26}
$$

$$
t_j^{l+1} = \|W_{j,:}^{l+1}\|_q, j = 1, ..., d. \tag{27}
$$

Since the problem (25) is separable in rows of W, solving it amounts to solving d independent sub-problems

$$
W_{j,:}^{l+1} = \arg\min_{W_{j,:}} \left\{ \frac{\rho}{2}\|W_{j,:}\|^2 + (-z_j^l)\|W_{j,:}\|_q - \langle U_{j,:}^l, W_{j,:}\rangle \right\} \tag{28}
$$

$W_{j,:}^{l+1}$ can be explicitly computed as showing in Appendix B. We refer to Appendix B for the calculation details.

Hence, DCA based algorithms for solving (22) with $q \in \{1, 2, \infty\}$ are described as follows.

DCA-$\ell_{q,0}$: DCA for solving (22) with $q \in \{1, 2, \infty\}$

Initialization: Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $\rho > L$ and $l \leftarrow 0$.

Repeat

1. Compute $(U^l, v^l, z^l) = \frac{1}{n}\sum_{i=1}^n (U_i^l, v_i^l, z_i^l)$, where (U_i^l, v_i^l, z_i^l), $i = 1, ..., n$ are defined in (24).

2. Compute $(W^{l+1}, b^{l+1}, t^{l+1})$ as follows

$$q = 1 : W^{l+1}_{j,:} = \left(|U^l_{j,:}|/\rho - (-z^l_j)/\rho\right)_+ \circ \text{sign}(U^l_{j,:})$$

$$q = 2 : W^{l+1}_{j,:} = \begin{cases} \left(1 - \frac{-z^l_j}{\|U^l_{j,:}\|_2}\right) U^l_{j,:}/\rho & \text{if } \|U^l_{j,:}\|_2 > -z^l_j \\ 0 & \text{if } \|U^l_{j,:}\|_2 \leq -z^l_j. \end{cases}$$

$$q = \infty : W^{l+1}_{j,:} = \begin{cases} U^l_{j,:}/\rho - \left(\frac{1}{-z^l_j}|U^l_{j,:}| - \delta\right)_+ \circ \text{sign}(U^l_{j,:}) & \text{if } \|U^l_{j,:}\|_1 > -z^l_j \\ 0 & \text{if } \|U^l_{j,:}\|_1 \leq -z^l_j, \end{cases} \tag{29}$$

where δ satisfies $\sum_{k=1}^{Q} \left(\frac{1}{-z^l_j}|U^l_{j,k}| - \delta\right)_+ = 1$.

3. $l \leftarrow l + 1$.

Until Stopping criterion.

3.2. SDCA for solving the problem (22)

In SDCA, at each iteration l, we have to compute $(U^l_i, v^l_i, z^l_i) \in \partial h_i(W^l, b^l, t^l)$ for $i \in s_l$ and keep $(U^l_i, v^l_i, z^l_i) = (U^{l-1}_i, v^{l-1}_i, z^{l-1}_i)$ for $i \notin s_l$, where s_l is a randomly chosen subset of the indices, and solve the convex sub-problem taking the form of (23). Hence, SDCA for solving (22) is described below.

SDCA-$\ell_{q,0}$: SDCA for solving (22) with $q \in \{1, 2, \infty\}$

Initialization: Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, $t^0_j = \|W^0_{j,:}\|_q$, $\rho > L$, $s_0 = \{1, ..., n\}$ and $l \leftarrow 0$.

Repeat

1. Compute (U^l_i, v^l_i, z^l_i) by (24) if $i \in s_l$ and keep $(U^l_i, v^l_i, z^l_i) = (U^{l-1}_i, v^{l-1}_i, z^{l-1}_i)$ if $i \notin s_l$. Set $(U^l, v^l, z^l) = \frac{1}{n}\sum_{i=1}^{n}(U^l_i, v^l_i, z^l_i)$.

2. Compute $(W^{l+1}, b^{l+1}, t^{l+1})$ according to step 2 of DCA.

3. $l \leftarrow l + 1$ and randomly choose a small subset $s_l \subset \{1, ..., n\}$.

Until Stopping criterion.

4. Numerical Experiment

We observe that all calculations of (U_i^l, v_i^l, z_i^l) and $(W^{l+1}, b^{l+1}, t^{l+1})$ in DCA and SDCA for the problem (22) are exactly defined in explicit form. Therefore the inexact version ISDCA is useless (Remark 2). Hence in our experiments we consider only SDCA.

4.1. Datasets

To evaluate the performance of algorithms, we performed numerical experiments on two types of data: real datasets (*covertype*, *madelon*, *miniboone*, *protein*, *sensit* and *sensorless*) and simulated datasets (*sim_1*, *sim_2* and *sim_3*). All real-world datasets are taken from the well-known UCI and LibSVM data repositories. The three synthetic datasets (*sim_1*, *sim_2* and *sim_3*) are generated by the same process proposed in Witten & Tibshirani (2011). In the first dataset (*sim_1*), variables are independent and have different means in each class. In dataset (*sim_2*), variables also have different means in each class, but they are dependent. The last synthetic dataset (*sim_3*) has different one-dimensional means in each class with independent variables.

More details of real-world datasets and the procedure for generating simulated dataset are given in Appendix D.

4.2. Comparative algorithms

We will compare the performance of the proposed SDCA with standard DCA as well as four other algorithms: the first two algorithms, named `SPGD-`$\ell_{2,1}$ and `msgl`, use the convex regularization $\ell_{2,1}$ instead of $\ell_{2,0}$ and deal with the following problem

$$\min_{W,b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, W, b) + \lambda \|W\|_{2,1} \right\}. \tag{30}$$

`SPGD-`$\ell_{2,1}$ is a version of stochastic gradient descent algorithm Bottou (1998); LeCun et al. (1998), the most popular stochastic algorithm in machine learning, while `msgl`

(Vincent & Hansen (2014)) is a coordinate gradient descent method. The last two algorithms, named nm−APG (Li & Lin (2015)) and DC−PN (Rakotomamonjy et al. (2016)), are the most recent which consider nonconvex regularization $\ell_{2,0}$. We refer to Appendix C for more details about these algorithms.

4.3. Experiment setting

We randomly split each dataset into a training set and a test set. The training set contains 80% of the total number of points and the remaining 20% are used as a test set.

We use the early-stopping condition for SDCA and SPGD−$\ell_{2,1}$. Early-stopping is a well-know technique in machine learning, especially in stochastic learning which permits to avoid the over-fitting in learning. More precisely, after each epoch, we compute the classification accuracy on a validation set which contains 20% randomly chosen data points of training set. We stop SDCA and SPGD−$\ell_{2,1}$ if the classification accuracy is not improved after $n_{patience} = 5$ epochs. The batch size of stochastic algorithms (SDCA and SPGD−$\ell_{2,1}$) is set to 10%. DCA is stopped if the difference between two consecutive objective functions is smaller than a threshold $\epsilon_{stop} = 10^{-6}$. For *msgl*, we use its default stopping parameters as in (Vincent & Hansen, 2014). We also stop algorithms if they exceed 2 hours of running time in the training process.

The parameter α for controlling the tightness of zero-norm approximation is chosen in the set $\{0.5, 1, 2, 5\}$. We use the solution-path procedure for the trade-off parameter λ. Let $\lambda_1 > \lambda_2 > ... > \lambda_l$ be a decreasing sequence of λ. At step k, we solve the problem (20) with $\lambda = \lambda_k$ from the initial point chosen as the solution of the previous step $k-1$. Starting with a large value of λ, we privilege the sparsity of solution (i.e. selecting very few variables) over the classification ability. Then by decreasing the value λ, we select more variables in order to increase the classification accuracy. In our experiments, the sequence of λ is set to $\{10^4, 3\times 10^3, 10^3, \ldots, 3\times 10^{-3}, 10^{-3}\}$.

All experiments are performed on a PC Intel (R) Xeon (R) E5-2630 v2 @2.60 GHz with 32GB RAM. In order to evaluate the performance of algorithms, we consider the following three criteria: the classification accuracy (percentage of well-classified points on the test set), the sparsity of obtained solutions and the running time (measured in

seconds). The sparsity is computed as the percentage of selected variables. Note that a variable $j \in \{1, \ldots, d\}$ is considered to be removed if all components of the row j of W are smaller than a threshold, i.e., $|W_{j,i}| \leq 10^{-8}, \forall i \in 1, \ldots, Q$. We perform each algorithm 10 times and report the average result over 10 runs.

Our experiment is composed of two parts.

Experiment 1: this is the main part of our experiment where we study the effectiveness of SDCA compared with other algorithms. For this purpose, among three regularizations $\ell_{q,0}$ and two approximations of the ℓ_0 norm we fix the $\ell_{2,0}$ regularization and the capped-ℓ_1 approximation. As the capped-ℓ_1 function is nonsmooth, the resulting approximate problem is nonsmooth (and nonconvex) (whereas the exponential approximation is a smooth function). We perform comparative experiments on SDCA-$\ell_{2,0}$-capℓ_1, DCA-$\ell_{2,0}$-capℓ_1, as well as the four comparative algorithms mentioned above. The comparative results (the average on 10 runs) are reported in Table 1. To simplify the presentation, in this experiment we use SDCA, DCA and SPGD instead of SDCA-$\ell_{2,0}$-capℓ_1, DCA-$\ell_{2,0}$-capℓ_1 and SPGD-$\ell_{2,1}$ respectively.

Experiment 2: for the purpose of feature selection, the effect of regularization terms as well as of approximation functions is an important matter. In this experiment we study the effectiveness of SDCA in terms of different non-convex regularizations $\ell_{q,0}$ ($q \in \{1, 2, +\infty\}$) by comparing three algorithms SDCA-$\ell_{1,0}$-exp, SDCA-$\ell_{2,0}$-exp and SDCA-$\ell_{\infty,0}$-exp (the exponential approximation is fixed for this experiment). The results are summarized in Table 2 (the columns 3–5). We also study the effect of the approximation functions (capped-ℓ_1 and exponential approximation) by comparing two algorithms: SDCA-$\ell_{2,0}$-exp and
SDCA-$\ell_{2,0}$-capℓ_1. The results are given in the same Table 2 (4th and 6th column).

4.4. Numerical results and Comments

Comments on Experiment 1.

Stochastic SDCA versus standard DCA:

As expected, in terms of classification accuracy, SDCA produces fairly similar results comparing with DCA, with a gap less than 1% on 8 out of 9 datasets. More

Table 1: Experiment 1: The average results on 10 runs of six algorithms on all datasets. Bold values correspond to the best results on each row. n, d and Q is the number of instances, the number of variables and the number of classes respectively. NA means that the algorithm fails to furnish a result after 12 hours.

Dataset	Performance	SDCA	DCA	SPGD	mgsl	mn-APG	DC-PN
covertype	accuracy%	70.40	70.40	66.97	71.22	NA	**71.43**
n=581,012	sparsity%	**57.41**	57.41	100	68.52	NA	93.21
d=54, Q=7	time (s)	**7.47**	61.15	60.59	525.49	NA	64.95
madelon	accuracy%	**62.18**	61.28	61.79	60.48	56.99	59.49
n=2,600	sparsity%	**0.40**	0.93	1.00	0.67	**0.40**	56.87
d=500, Q=2	time (s)	**0.15**	0.21	1.07	23.92	2.43	0.55
miniboone	accuracy%	83.31	83.74	83.86	81.99	**83.95**	83.57
n=130,065	sparsity%	**6.00**	**6.00**	11.00	10.00	**6.00**	14.00
d=50, Q=2	time (s)	**1.18**	7.04	8.77	121.17	56.51	7.96
protein	accuracy%	66.41	67.04	66.59	67.34	67.04	**67.39**
n=24,387	sparsity%	**22.64**	50.47	92.70	47.15	44.29	63.30
d=357, Q=3	time (s)	**1.13**	3.35	11.73	5.59	31.04	4.13
sensit	accuracy%	78.59	78.92	**79.52**	79.02	79.16	79.20
n=98,528	sparsity%	33.80	56.67	27.00	23.00	**14.33**	50.33
d=100, Q=3	time (s)	**2.94**	26.36	22.44	11.16	332.58	10.42
sensorless	accuracy%	84.77	**89.60**	86.07	85.06	89.09	87.66
n=58,509	sparsity%	68.06	53.47	88.89	50.00	**36.81**	97.22
d=48, Q=11	time (s)	**2.45**	24.75	8.16	199.00	360.02	30.11
sim_1	accuracy%	72.24	72.24	71.48	**72.33**	71.02	72.18
n=100,000	sparsity%	**80.00**	**80.00**	83.50	82.00	**80.00**	96.00
d=50, Q=4	time (s)	0.50	**0.30**	7.16	214.83	344.65	2.77
sim_2	accuracy%	**68.50**	67.70	67.62	68.42	66.96	67.50
n=150,000	sparsity%	**80.00**	**80.00**	82.00	82.00	**80.00**	88.00
d=50, Q=3	time (s)	**1.02**	4.29	7.77	367.29	319.28	3.93
sim_3	accuracy%	99.69	**99.88**	99.70	99.93	99.86	99.85
n=250,000	sparsity%	**80.00**	**80.00**	**80.00**	80.20	**80.00**	80.13
d=500, Q=4	time (s)	**21.95**	249.74	212.71	1581.44	571.19	144.72

precisely, the gap is zero in two datasets, in favor of SDCA in 2 datasets, and in favor of DCA in 5 datasets. For the remaining dataset *sensorless* the gap is 4.83% in favor of DCA.

As for the sparsity of solution, the two algorithms give the same result on 5 out of 9 datasets, SDCA is better than DCA on 2 datasets (the gap is 27.83% and 22.87%), and DCA is better than SDCA on 2 datasets (the gap is 0.53% and 14.59%).

Concerning the running time, not surprisingly, SDCA is much faster than DCA, the ratio of gain varies from 1.41 (*madelon*) to 11.38 (*sim_3*) times on 8 out of 9 datasets. Except for *sim_1* where DCA is slightly faster than SDCA (0.5 versus 0.3 seconds).

Overall, SDCA achieves similar solutions' quality in a much shorter time than DCA, that is the aim of our work.

SDCA and DCA versus other algorithms.

In terms of classification accuracy, SDCA and DCA produce fairly similar results compared with the best results given by the four algorithms SPGD, mgsl, mn-APG, DC-PN with a gap less than 1% on 8 out of 9 datasets (4 in favor of SDCA/DCA, 4 in favor of the best among the four other algorithms) and equal to 1.03% in the remaining dataset (*covertype*). More concretely, comparing with the two $\ell_{2,1}$ (convex) regularization algorithms SPGD and mgsl, the gap versus SPGD (resp. mgsl) is in favor of SDCA/DCA) on 6 (resp. 5) datasets. As for the two $\ell_{2,0}$ (nonconvex) regularization algorithms mn-APG, DC-PN, the gap versus mn-APG as well as DC-PN is in favor of SDCA/DCA on 6 datasets, and the gap versus mn-APG is zero on 1 dataset.

Regarding the sparsity, SDCA gives the sparsest solutions on 7 datasets, while DCA as well as mn-APG get the best result on 5 datasets (these three algorithms furnish the same sparsity of solutions on 4 datasets). The gain of SDCA versus the three remaining algorithms is considerable. It varies from 0.68% (resp. 0.27%) to 70.06% (resp. 24.51%) comparing with SPGD (resp. mgsl). Relating to DC-PN, the gap varies from 16% to 56.47% on 6 datasets, and from 0.13% to 8% on 3 datasets. Altogether, SDCA is the best while DC-PN is the worst.

Overall, combining the two criteria - accuracy and sparsity, SDCA is the best, followed by DCA, and then mn-APG (not counting *covertype* for which mn-APG is failed after

12 hours), while DC−PN is the worse.

In the matter of rapidity (running time), SDCA (resp. DCA) is the fastest on 8 (resp. 1) dataset(s). The gain of SDCA versus the four other algorithms is huge. The ratio of gain varies from 3.33 (resp. 4.95) to 14.32 (resp. 360.09) times comparing with SPGD (resp. mgsl), and from 16.20 (resp. 3.65) to 689.30 (resp. 12.29) times compared with mn−APG (resp. DC−PN). Altogether, SDCA is the fastest algorithm, followed by DCA, and then DC−PN, while mn−APG is the slowest, in particular it fails to get a solution after 12 hours on *covertype*.

In summary, as expected, SDCA reduces considerably the running time of DCA while achieving equivalent classification accuracy and better sparsity. Moreover, SDCA outperforms the four related algorithms SPGD, msgl, mn−APG and DC−PN. Hence, SDCA is the best algorithm on both quality (accuracy and sparsity) and rapidity.

Comments on Experiment 2.

SDCA with three regularizations $\ell_{1,0}$, $\ell_{2,0}$, $\ell_{\infty,0}$:

In terms of classification accuracy, the three algorithms SDCA−$\ell_{1,0}$−exp, SDCA−$\ell_{2,0}$−exp and SDCA−$\ell_{\infty,0}$−exp get similar results with a gap less than 1% on all datasets. In particular, the gap between SDCA−$\ell_{1,0}$−exp and SDCA−$\ell_{2,0}$−exp is lower than 0.3% on 6 datasets. The gain is in favor of SDCA−$\ell_{1,0}$ on 4 datasets, of SDCA−$\ell_{\infty,0}$−exp on 4 datasets (they have the same best result on *sim_1* and of SDCA−$\ell_{2,0}$−exp on 2 datasets.

As for the sparsity of solution, SDCA−$\ell_{2,0}$−exp is the best on 7 datasets (except for *covertype* and *protein*), while SDCA−$\ell_{\infty,0}$−exp is the worse on 8 datasets (except for *covertype* it gives the best result). The gain of SDCA−$\ell_{2,0}$−exp versus SDCA−$\ell_{1,0}$−exp (resp. SDCA−$\ell_{\infty,0}$−exp) varies from 0.25% to 17.19% (resp. from 0.30% to 60.42%), and the gain of SDCA−$\ell_{1,0}$−exp versus SDCA−$\ell_{\infty,0}$−exp varies from 0.05% to 43.23%.

Overall, combining the two criteria - accuracy and sparsity, SDCA−$\ell_{1,0}$−exp and SDCA−$\ell_{2,0}$−exp realize a better trade-off between classification and sparsity of solution than SDCA−$\ell_{\infty,0}$−exp, and SDCA−$\ell_{2,0}$−exp is the best.

In terms of rapidity, the three algorithms are comparable on 6 datasets where the running time is less than 1.5 second. As for the 3 remaining datasets, each algorithm

Table 2: The average results on 10 runs of Experiment 2 (SDCA with different regularization $\ell_{q,0}$, $q \in \{1, 2, +\infty\}$) and Experiment 3 ($\ell_{2,0}$ regularization with exponential/cap1 approximations). Bold values correspond to the best results on each row.

Dataset	Performance	$\ell_{1,0}$-exp	$\ell_{2,0}$-exp	$\ell_{\infty,0}$-exp	$\ell_{2,0}$-cap1
covertype	accuracy%	71.34	**71.62**	69.92	70.40
	sparsity%	69.91	61.11	60.49	**57.41**
	time (s)	10.27	**4.74**	11.93	7.47
madelon	accuracy%	61.92	**62.12**	61.68	62.18
	sparsity%	0.65	**0.40**	0.70	**0.40**
	time (s)	**0.14**	0.16	0.16	0.15
miniboone	accuracy%	**83.90**	83.84	83.10	83.31
	sparsity%	8.00	**6.00**	8.00	**6.00**
	time (s)	1.57	3.60	1.62	**1.18**
protein	accuracy%	67.23	67.84	**68.13**	66.41
	sparsity%	63.67	64.89	92.79	**22.64**
	time (s)	1.47	1.28	1.36	**1.13**
sensit	accuracy%	79.64	78.67	**79.73**	78.59
	sparsity%	34.00	**28.33**	53.67	33.80
	time (s)	3.11	3.48	**1.61**	2.94
sensorless	accuracy%	**87.33**	86.52	86.69	84.77
	sparsity%	54.69	**37.50**	97.92	68.06
	time (s)	**1.40**	1.47	1.41	2.25
sim_1	accuracy%	**72.24**	72.22	**72.24**	**72.24**
	sparsity%	**80.00**	**80.00**	**80.00**	**80.00**
	time (s)	**0.46**	**0.46**	0.56	0.50
sim_2	accuracy%	68.48	68.53	**68.71**	68.50
	sparsity%	**80.00**	**80.00**	**80.00**	**80.00**
	time (s)	**0.73**	0.79	0.97	1.02
sim_3	accuracy%	**99.93**	99.69	99.56	**99.69**
	sparsity%	**80.00**	**80.00**	80.73	**80.00**
	time (s)	**10.74**	36.61	22.11	21.45

is winner on 1 dataset. SDCA-$\ell_{2,0}$-exp is the winner on *covertype* with the ratio of gain being 2.17 and 2.52 versus SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{1,0}$-exp respectively. SDCA-$\ell_{1,0}$-exp is 3.4 (resp. 2.06) times faster than SDCA-$\ell_{2,0}$-exp (resp. SDCA-$\ell_{\infty,0}$-exp) on *sim_3* while SDCA-$\ell_{\infty,0}$-exp is 1.93 (resp. 2.16) times faster than SDCA-$\ell_{1,0}$-exp (resp. SDCA-$\ell_{2,0}$-exp) on *sensit*.

SDCA-$\ell_{1,0}$-exp is the fastest and SDCA-$\ell_{2,0}$-exp is the slowest among the three algorithms. SDCA-$\ell_{1,0}$-exp is up to 3.4 time faster than SDCA-$\ell_{2,0}$-exp and 2.06 times faster than SDCA-$\ell_{\infty,0}$-exp.

In summary, the two algorithms SDCA-$\ell_{1,0}$-exp and SDCA-$\ell_{2,0}$-exp are comparable and they are better than SDCA-$\ell_{\infty,0}$-exp on both quality and rapidity, in particular in terms of sparsity. Hence, for feature selection purpose, it is suggested to use the first two algorithms.

SDCA with the two approximations - exponential and capℓ_1 functions:

In terms of classification accuracy SDCA-$\ell_{2,0}$-exp is slightly better than SDCA-$\ell_{2,0}$-capℓ_1 on 8 datasets with a gap less than 1% on 5 datasets and less than 1.8% on 3 datasets. For the remain dataset (*sim_1*) the gain is only 0.02% in favor of SDCA-$\ell_{2,0}$-capℓ_1.

Regarding sparsity, the two algorithms get the same result on 5 datasets, SDCA-$\ell_{2,0}$-capℓ_1 is winner on *covertype* and *protein* with the gain 3.7% and 42.25% respectively, while SDCA-$\ell_{2,0}$-exp is winner on *sensit* and *sensorless* with the gain 5.47% and 43.56% respectively.

As for running time, SDCA-$\ell_{2,0}$-exp is faster than SDCA-$\ell_{2,0}$-capℓ_1 from 1.08 to 1.57 times on 4 datasets, while the later is faster than the former from 1.13 to 3.05 times on 5 datasets.

Overall, SDCA-$\ell_{2,0}$-exp seems to be better for the purpose of classification, and it realizes a better trade-off between quality and rapidity in most cases.

5. Conclusion

We have proposed two novel DCA-based algorithms, stochastic DCA and inexact stochastic DCA for minimizing a large sum of DC functions, with the aim of reducing

the computation cost of DCA in large-scale setting. The sum structure of the objective function F permits us to work separately on the component functions F_i. The stochastic DCA is then proposed to tackle problems with huge numbers of F_i while the inexact stochastic DCA aims to address large-scale setting and big data. We have carefully studied the convergence properties of the proposed algorithms. It turns out that the convergence to a critical point of both stochastic DCA and inexact stochastic DCA is guaranteed with probability 1. Furthermore, we have developed DCA and SDCA to group variables selection in multi-class logistic regression, an important problem in machine learning. By using a suitable DC decomposition of the objective function we have designed a DCA scheme in which all computations are explicit and inexpensive. Consequently SDCA is very inexpensive. Numerical results showed that, as expected, `SDCA` reduces considerably the running time of standard `DCA` while achieving equivalent classification accuracy and better sparsity. Moreover, `SDCA` outperforms the four related algorithms on both quality and rapidity, and the gain is considerable. We are convinced that SDCA is an efficient variant of DCA, especially for large-scale setting and Big data.

Continuing this research direction, in future works we will develop novel versions of DCA-based algorithms (e.g. online/stochastic/approximate/like DCA) for other problems in order to accelerate the convergence of DCA and to deal with large-scale setting and Big data.

Acknowledgment

This research is funded by Foundation for Science and Technology Development of Ton Duc Thang University (FOSTECT), website: http://fostect.tdtu.edu.vn, under Grant FOSTECT.2017.BR.10.

Appendix A. Proof of Theorem 2

To prove Theorem 2, we will use the following lemma.

Lemma 1. *Let $f : \mathbb{R}^d \to \mathbb{R} \cap \{+\infty\}$ be a ρ-convex function. For any $\epsilon \geq 0$ and any $v \in \partial_\epsilon f(x)$ with $x \in$ dom f, we have*

$$2\epsilon + f(y) \geq f(x) + \langle v, y - x\rangle + \frac{\rho}{4}\|y - x\|^2, \ \forall y \in \mathbb{R}^d.$$

Proof. Since $v \in \partial_\epsilon f(x)$, we have

$$\epsilon + f(z) \geq f(x) + \langle v, z - x\rangle, \ \forall z \in \mathbb{R}^d.$$

Replacing z with $x + t(y - x)$ in this inequality gives that

$$\epsilon + f(x + t(y - x)) \geq f(x) + t\langle v, y - x\rangle, \ \forall y \in \mathbb{R}^d.$$

It follows from the ρ-convexity of f that for $y \in \mathbb{R}^d$ and $t \in (0, 1)$,

$$tf(y) + (1 - t)f(x) \geq f(x + t(y - x)) + \frac{\rho t(1 - t)}{2}\|y - x\|^2.$$

Summing the two above inequalities gives us

$$\epsilon + tf(y) \geq tf(x) + t\langle v, y - x\rangle + \frac{\rho t(1 - t)}{2}\|y - x\|^2.$$

Thus, the conclusion follows from this inequality with $t = 1/2$. □

Proof. (of Theorem 2) a) Let x_i^0 be the copies of x^0. We set $x_i^{l+1} = x^{l+1}$ for all $i \in s_{l+1}$ and $x_j^{l+1} = x_j^l$ for $j \notin s_{l+1}$. Set $\epsilon_i^0 = \epsilon^0$ and $\epsilon_i^{l+1} = \epsilon^{l+1}$ if $i \in s_{l+1}$, ϵ_i^l otherwise. We then have $v_i^l \in \partial_{\epsilon_i^l} h_i(x_i^l)$ for $i = 1, ..., n$. Let T_i^l be the function given by

$$T_i^l(x) = g_i(x) - h_i(x_i^l) - \left\langle x - x_i^l, v_i^l \right\rangle + 2\epsilon_i^l.$$

It follows from $v_i^l \in \partial_{\epsilon_i^l} h_i(x_i^l)$ that

$$\epsilon_i^l + h_i(x) \geq h_i(x_i^l) + \left\langle x - x_i^l, v_i^l \right\rangle.$$

This implies $T_i^l(x) \geq F_i(x) + \epsilon_i^l \geq F_i(x)$ for all $l \geq 0$, $i = 1, ..., n$. We also observe that x^{l+1} is an ϵ^l-solution of the following convex problem

$$\min_x T^l(x) := \frac{1}{n}\sum_{i=1}^n T_i^l(x) \tag{A.1}$$

Therefore

$$\begin{aligned} T^l(x^{l+1}) \leq T^l(x^l) + \epsilon^l &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[T_i^l(x^l) - T_i^{l-1}(x^l)] + \epsilon^l \\ &= T^{l-1}(x^l) + \frac{1}{n}\sum_{i\in s_l}[F_i(x^l) + 2\epsilon^l - T_i^{l-1}(x^l)] + \epsilon^l, \end{aligned} \tag{A.2}$$

where the second equality follows from $T_i^l(x^l) = F_i(x^l) + 2\epsilon^l$ for all $i \in s_l$. Let $\mathcal{F}_l$ denote the σ-algebra generated by the entire history of ISDCA up to the iteration l, i.e., $\mathcal{F}_0 = \sigma(x^0, \epsilon^0)$ and $\mathcal{F}_l = \sigma(x^0, ..., x^l, \epsilon^0, ..., \epsilon^l, s_0, ..., s_{l-1})$ for all $l \geq 1$. By taking the expectation of the inequality (A.2) conditioned on $\mathcal{F}_l$, we have

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{n}\left[T^{l-1}(x^l) - F(x^l)\right] + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Since $\sum_{l=0}^{\infty}\epsilon_i^l < +\infty$ with probability 1, by applying the supermartingale convergence theorem (Neveu, 1975; Bertsekas et al., 2003) to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}, \{\frac{b}{n}[T^{l-1}(x^l) - F(x^l)]\}$ and $\{(\frac{2b}{n} + 1)\epsilon^l\}$, we conclude that the sequence $\{T^{l-1}(x^l, y^l) - \alpha^*\}$ converges to $T^* - \alpha^*$ and

$$\sum_{l=1}^{\infty}\left[T^{l-1}(x^l) - F(x^l)\right] < \infty, \tag{A.3}$$

with probability 1. Therefore $\{F(x^l)\}$ converges almost surely to T^*.

b) By $v_i^{l-1} \in \partial_{\epsilon_i^{l-1}} h_i(x_i^{l-1})$ and Lemma 1, we have

$$2\epsilon_i^{l-1} + h_i(x) \geq h_i(x_i^{l-1}) + \langle x - x_i^{l-1}, v_i^{l-1}\rangle + \frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2, \ \forall x \in \mathbb{R}^d.$$

This implies

$$F_i(x) \leq T_i^{l-1}(x) - \frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2. \tag{A.4}$$

From (A.2) and (A.4) with $x = x^l$, we have

$$T^l(x^{l+1}) \leq T^{l-1}(x^l) - \frac{1}{n}\sum_{i\in s_l}\frac{\rho(h_i)}{4}\|x - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l. \tag{A.5}$$

Taking the expectation of the inequality (A.5) conditioned on $\mathcal{F}_l$, we obtain

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b}{4n^2}\sum_{i=1}^{n}\rho(h_i)\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n} + 1\right)\epsilon^l.$$

Combining this and $\rho = \min_{i=1,\ldots,n} \rho(h_i) > 0$ gives us

$$\mathbb{E}\left[T^l(x^{l+1})|\mathcal{F}_l\right] \leq T^{l-1}(x^l) - \frac{b\rho}{4n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 + \left(\frac{2b}{n}+1\right)\epsilon^l.$$

Applying the supermartingale convergence theorem to the nonnegative sequences $\{T^{l-1}(x^l) - \alpha^*\}$, $\{\frac{b}{4\rho n^2}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2\}$ and $\{(\frac{2b}{n}+1)\epsilon^l\}$, we get

$$\sum_{l=1}^{\infty}\sum_{i=1}^{n}\|x^l - x_i^{l-1}\|^2 < \infty,$$

with probability 1. In particular, for $i = 1, ..., n$, we have

$$\sum_{l=1}^{\infty}\|x^l - x_i^{l-1}\|^2 < \infty, \tag{A.6}$$

and hence $\lim_{l\to\infty}\|x^l - x_i^{l-1}\| = 0$ almost surely.

c) Assume that there exists a sub-sequence $\{x^{l_k}\}$ of $\{x^l\}$ such that $x^{l_k} \to x^*$ almost surely. From (A.6), we have $\|x^{l_k+1} - x_i^{l_k}\| \to 0$ almost surely. Without loss of generality, we can suppose that the sub-sequence $v_i^{l_k} \to v_i^*$ almost surely. From the proof of (a), we have

$$\frac{1}{n}\sum_{i=1}^{n}\epsilon_i^l \leq T^l(x^{l+1}) - F(x^{l+1}).$$

From this and (A.3) it follows that ϵ_i^l converges to 0 as $l \to +\infty$ with probability 1. Since $v_i^{l_k} \in \partial_{\epsilon_i^{l_k}} h_i(x_i^{l_k})$, $\epsilon_i^{l_k} \to 0$ with probability 1, and by the closedness property of the ϵ-subdifferential mapping $\partial_{\epsilon_i^{l_k}} h_i$, we have $v_i^* \in \partial h_i(x^*)$. Since x^{l_k+1} is a ϵ^{l_k}-solution of the problem $\min_x T^{l_k}(x)$, we obtain

$$T^{l_k}(x^{l_k+1}) \leq T^{l_k}(x) + \epsilon^{l_k}. \tag{A.7}$$

Taking $k \to \infty$ gives us

$$\limsup_{l_k\to+\infty} G(x^{l_k+1}) \leq G(x) - \langle x - x^*, v^*\rangle,\ \forall x \in \mathbb{R}^d,$$

with probability 1, where $v^* = \frac{1}{n}\sum_{i=1}^{n} v_i^* \in \partial H(x^*)$ almost surely. It follows from this with $x = x^*$ that

$$\limsup_{l_k\to+\infty} G(x^{l_k+1}) \leq G(x^*),$$

almost surely. Combining this with the lower semi-continuity of G gives us that

$$\lim_{l_k \to +\infty} G(x^{l_k+1}) = G(x^*),$$

almost surely. Thus, we have

$$G(x^*) \leq G(x) - \langle x - x^*, v^* \rangle, \ \forall x \in \mathbb{R}^d,$$

almost surely. This implies

$$v^* \in \partial G(x^*), \tag{A.8}$$

with probability one. Therefore,

$$v^* \in \partial G(x^*) \cap \partial H(x^*), \tag{A.9}$$

with probability 1. This implies that x^* is a critical point of F with probability 1 and the proof is then complete. □

Appendix B. Calculation details of $W_{j,:}^{l+1}$ in DCA

From (28) we can write

$$W_{j,:}^{l+1} = \arg\min_{W_{j,:}} \left\{ \frac{1}{2} \|W_{j,:} - U_{j,:}^l/\rho\|^2 + (-z_j^l) \|W_{j,:}\|_q \right\}.$$

$$W_{j,:}^{l+1} = \mathbf{prox}_{(-z_j^l)/\rho \|\cdot\|_q} \left(U_{j,:}^l/\rho \right),$$

where the proximal operator $\mathbf{prox}_f(\nu)$ is defined by

$$\mathbf{prox}_f(\nu) = \arg\min_t \left\{ \frac{1}{2} \|t - \nu\|^2 + f(t) \right\}.$$

The proximal operator of $(-z_j^l)/\rho \| \cdot \|_q$ can be efficiently computed (Parikh & Boyd, 2014). The computation of $\mathbf{prox}_{(-z_j^l)/\rho \|\cdot\|_q} (\nu/\rho)$ can be summarized in Table B.3.

Appendix C. About comparative algorithms

Appendix C.1. Stochastic Proximal Gradient Descent algorithm

SPGD-$\ell_{2,1}$: Stochastic Proximal Gradient Descent for solving (30)

Table B.3: Computation of $W_{j,:}^{l+1} = \mathbf{prox}_{(-z_j^l)/\rho\|.\|_q}\left(U_{j,:}^l/\rho\right)$ corresponding to $q \in \{1, 2, \infty\}$.

q	$\mathbf{prox}_{(-z_j^l)/\rho\|.\|_q}\left(U_{j,:}^l/\rho\right)$
1	$\left(\lvert U_{j,:}^l\rvert/\rho - (-z_j^l)/\rho\right)_+ \circ \text{sign}(U_{j,:}^l)$
2	$\begin{cases} \left(1 - \frac{-z_j^l}{\|U_{j,:}^l\|_2}\right) U_{j,:}^l/\rho & \text{if } \|U_{j,:}^l\|_2 > -z_j^l \\ 0 & \text{if } \|U_{j,:}^l\|_2 \leq -z_j^l. \end{cases}$
∞	$\begin{cases} U_{j,:}^l/\rho - \left(\frac{1}{-z_j^l}\lvert U_{j,:}^l\rvert - \delta\right)_+ \circ \text{sign}(U_{j,:}^l) & \text{if } \|U_{j,:}^l\|_1 > -z_j^l \\ 0 & \text{if } \|U_{j,:}^l\|_1 \leq -z_j^l, \end{cases}$ where δ satisfies $\sum_{k=1}^Q \left(\frac{1}{-z_j^l}\lvert U_{j,k}^l\rvert - \delta\right)_+ = 1$.

Initialization: Choose $(W^0, b^0) \in \mathbb{R}^{d\times Q} \times \mathbb{R}^Q$, and $l \leftarrow 0$.

Repeat

1. Randomly choose a small subset $s_l \subset \{1, ..., n\}$. Set $\alpha_l = \frac{n}{10l}$. Compute $\bar{U}_{:,k}^l = W_{:,k}^l - \frac{\alpha_l}{|s_l|}\sum_{i\in s_l}\left(p_k^l(x_i) - \delta_{ky_i}\right)x_i, k = 1, ...Q$.

2. Compute (W^{l+1}, b^{l+1}) by

$$\begin{aligned} W_{j,:}^{l+1} &= \left(\|\bar{U}_{j,:}^l\|_2 - \alpha_l\lambda\right)_+ \frac{\bar{U}_{j,:}^l}{\|\bar{U}_{j,:}^l\|_2}, j = 1, ..., d \\ b_k^{l+1} &= b_k^l - \frac{\alpha_l}{|s_l|}\sum_{i\in s_l}\left(p_k^l(x_i) - \delta_{ky_i}\right), k = 1, ..., Q. \end{aligned} \tag{C.1}$$

3. $l \leftarrow l + 1$.

Until Stopping criterion.

Appendix C.2. `msgl`

`msgl` is a coordinate gradient descent proposed in Vincent & Hansen (2014) to solve the problem (30).

Appendix C.3. Non-monotone Accelerated Proximal Gradient algorithm (nm−APG)

nm−APG Li & Lin (2015) is an accelerated proximal gradient based method for minimizing $f(x)+r(x)$ where $f(x)$ is a differentiable function with L-Lipschitz gradient and $r(x)$ is a nonconvex function. nm−APG requires to compute the proximal mapping of the DC function η_α. However, this proximal mapping does not have a closed form. We therefore use DCA to compute the proximal mapping of η_α in nm−APG.

Appendix C.4. DC-Proximal Newton algorithm (DC-PN)

DC−PN was proposed in Rakotomamonjy et al. (2016) for minimizing $f(x)+h(x)$, where $f(x) = f_1(x)-f_2(x)$ is a DC function, twice differentiable, with $f(x)$ verifying the L-Lipschitz gradient property; $h(x) = h_1(x) - h_2(x)$ where both h_1 and h_2 are convex functions and (possibly) non-differential. Note that, in DC−PN, L-BFGS is employed for approximating the Hessian matrix H_k; and the sub-problem is solved by minFunc solver [1].

Appendix D. Datasets in numerical experiments

covertype belongs to the Forest Cover Type Prediction from strictly cartographic variables challenge[2]. It is a very large dataset containing $581,012$ points described by 54 variables.

madelon is one of five datasets used in the NIPS 2003 feature selection challenge[3]. The dataset contains 2600 points, each point is represented by 500 variables. Among 500 variables, there are only 5 informative variables and 15 redundant variables (which are created by linear combinations of 5 informative variables). The 480 others variables were added and have no predictive power. Notice that *madelon* is a highly non-linear dataset.

[1] minFunc: unconstrained differentiable multivariate optimization in Matlab. http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html

[2] https://archive.ics.uci.edu/ml/datasets/Covertype

[3] https://archive.ics.uci.edu/ml/datasets/Madelon

miniboone is taken form the MiniBooNE experiment to observe neutrino oscillations[4], containing $130,065$ data points.

protein [5] is a dataset for classifying protein second structure state (α, β, and coil) of each residue in amino acid sequences, including $24,387$ data points.

sensit [5] dataset obtained from distributed sensor network for vehicle classification. It consists of $98,528$ data points categorized into 3 classes: Assault Amphibian Vehicle (AAV), Dragon Wagon (DW) and noise.

sensorless measures electric current drive signals from different operating conditions, which is classified into 11 different classes [6]. It is a huge dataset, which contains $58,509$ data points, described by 48 variables.

For *sim_1*: we generate a four-classes classification problem. Each class is assumed to have a multivariate normal distribution $\mathcal{N}(\mu_k, I)$, $k = 1, 2, 3, 4$ with dimension of $d = 50$. The first 10 components of μ_1 are 0.5, $\mu_{2j} = 0.5$ if $11 \leq j \leq 20$, $\mu_{3j} = 0.5$ if $21 \leq j \leq 30$, $\mu_{4j} = 0.5$ if $31 \leq j \leq 40$ and 0 otherwise. We generate $250,000$ instances with equal probabilities.

For *sim_2*: this synthetic dataset contains three classes of multivariate normal distributions $\mathcal{N}(\mu_k, \Sigma)$, $k = 1, 2, 3$, each of dimension $d = 50$. The components of $\mu_1 = 0$, $\mu_{2j} = 0.4$ and $\mu_{3j} = 0.8$ if $j \leq 40$ and 0 otherwise. The covariance matrix Σ is the block diagonal matrix with five blocks of dimension 10×10 whose element (j, j') is $0.6^{|j-j'|}$. We generate $150,000$ instances.

For *sim_3*: this synthetic dataset consists of four classes. For class $k = 1, 2, 3, 4$, $i \in C_k$ then $X_{ij} \sim \mathcal{N}(0, 1)$ for $j > 100$, and $X_{ij} \sim \mathcal{N}(\frac{k-1}{3}, 1)$ otherwise, where $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2. We generate $62,500$ data points for each class.

[4] https://archive.ics.uci.edu/ml/datasets/MiniBooNE+particle+identification

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

[6] https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis

References

Allen-Zhu, Z., & Yuan, Y. (2016). Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* ICML'16 (pp. 1080–1089).

Bagley, S. C., White, H., & Golomb, B. A. (2001). Logistic regression in the medical literature: Standards for use and reporting, with particular attention to one medical domain. *Journal of Clinical Epidemiology*, *54*, 979–985.

Bertsekas, D., Nedic, A., & Ozdaglar, A. (2003). *Convex analysis and optimization*. Athena Scientific.

Bertsekas, D. P. (2010). *Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey*. Technical Report Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Bertsekas, D. P. (2011). Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, *129*, 163–195.

Bottou, L. (1998). On-line learning and stochastic approximations. In D. Saad (Ed.), *On-line Learning in Neural Networks* (pp. 9–42). New York, NY, USA: Cambridge University Press.

Boyd, C. R., Tolson, M. A., & Copes, W. S. (1987). Evaluating trauma care: The TRISS method. Trauma Score and the Injury Severity Score. *The Journal of Trauma*, *27*, 370–378.

Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference (ICML 98)* (pp. 82–90). Morgan Kaufmann.

Cox, D. (1958). The regression analysis of binary sequences (with discussion). *J Roy Stat Soc B*, *20*, 215–242.

Defazio, A., Bach, F., & Lacoste-Julien, S. (2014a). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of Advances in Neural Information Processing Systems*.

Defazio, A., Caetano, T., & Domke, J. (2014b). Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the* 31^{st} *International Conference on Machine Learning*.

Genkin, A., Lewis, D. D., & Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Technometrics*, *49*, 291–304.

Healy, K., & Schruben, L. W. (1991). Retrospective simulation response optimization. In *1991 Winter Simulation Conference Proceedings.* (pp. 901–906).

Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26* (pp. 315–323). Curran Associates Inc.

Kim, J., Kim, Y., & Kim, Y. (2008). A Gradient-Based Optimization Algorithm for LASSO. *Journal of Computational and Graphical Statistics*, *17*, 994–1009.

King, G., & Zeng, L. (2001). Logistic Regression in Rare Events Data. *Political Analysis*, *9*, 137–163.

Le Thi, H. A., Le, H. M., Nguyen, V. V., & Pham Dinh, T. (2008). A DC programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, *2*, 259–278.

Le Thi, H. A., Le, H. M., Phan, D. N., & Tran, B. (2017). Stochastic DCA for the Large-sum of Non-convex Functions Problem and its Application to Group Variable Selection in Classification. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 3394–3403). volume 70.

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. (2018). DC programming and DCA: thirty years of developments. *Mathematical Programming*, (pp. 1–64).

Le Thi, H. A., Pham Dinh, T., Le, H. M., & Vo, X. T. (2015). DC approximation approaches for sparse optimization. *Eur. J. Oper. Res.*, *244*, 26–46.

Le Thi, H. A., & Phan, D. N. (2016). DC Programming and DCA for Sparse Optimal Scoring Problem. *Neurocomput.*, *186*, 170–181.

Le Thi, H. A., Phan, D. N., & Pham, D. T. (2019). DCA based approaches for bi-level variable selection and application for estimating multiple sparse covariance matrices. *Revised version Neurocomputing*, .

Le Thi (Home Page), H. A. (2005). DC Programming and DCA - Website of Le Thi Hoai An. http://www.lita.univ-lorraine.fr/~lethi/index.php/en/research/dc-programming-and-dca.html.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient backprop. In *Neural Networks: Tricks of the Trade* (pp. 9–50). Berlin, Heidelberg: Springer Berlin Heidelberg.

Li, H., & Lin, Z. (2015). Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems* (pp. 377–387).

Liao, J. G., & Chin, K.-V. (2007). Logistic regression for disease classification using microarray data: Model selection in a large p and small n case. *Bioinformatics*, *23*, 1945–1951.

Mairal, J. (2015). Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, *25*, 829–855.

Neveu, J. (1975). *Discrete-Parameter Martingales* volume 10 of *North-Holland Mathematical Library*. Elsevier.

Parikh, N., & Boyd, S. (2014). Proximal algorithms. *Found. Trends Optim.*, *1*, 127–239.

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

Pham Dinh, T., & Le Thi, H. A. (1998). A D. C. Optimization Algorithm for Solving the Trust-Region Subproblem. *SIAM Journal of Optimization*, *8*, 476–505.

Pham Dinh, T., & Le Thi, H. A. (2014). Recent advances in DC programming and DCA. *Transactions on Computational Collective Intelligence*, *8342*, 1–37.

Pham Dinh, T., & Souad, E. B. (1986). Algorithms for Solving a Class of Nonconvex Optimization Problems. Methods of Subgradients. In J. B. Hiriart-Urruty (Ed.), *North-Holland Mathematics Studies* (pp. 249–271). North-Holland volume 129 of *Fermat Days 85: Mathematics for Optimization*.

Phan, D. N., Le Thi, H. A., & Pham Dinh, T. (2017). Sparse covariance matrix estimation by DCA-Based Algorithms. *Neural Computation*, *29*, 3040–3077.

Phan, D. N., & Thi, H. A. L. (2019). Group variable selection via $\ell p, 0$ regularization and application to optimal scoring. *Neural Networks*, . doi:https://doi.org/10.1016/j.neunet.2019.05.011.

Rakotomamonjy, A., Flamary, R., & Gasso, G. (2016). DC Proximal Newton for Nonconvex Optimization Problems. *IEEE Transactions on Neural Networks and Learning Systems*, *27*, 636–647. doi:10.1109/TNNLS.2015.2418224.

Reddi, S. J., Sra, S., Poczos, B., & Smola, A. J. (2016). Proximal stochastic methods for Nonsmooth Nonconvex Finite-Sum Optimization. In *Advances in Neural Information Processing Systems* (pp. 1145–1153).

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Schmidt, M., Le Roux, N., & Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, *162*, 83–112.

Shalev-Schwartz, S., & Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, *14*, 567–599.

Subasi, A., & Erçelebi, E. (2005). Classification of EEG signals using neural network and logistic regression. *Comput. Methods Programs Biomed.*, *78*, 87–99.

Vincent, M., & Hansen, N. R. (2014). Sparse group lasso and high dimensional multinomial classification. *Comput. Stat. Data Anal.*, *71*, 771–786.

Witten, D. M., & Tibshirani, R. (2011). Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society: Series B*, *73*, 753–772.