SOLVING NON-CONVEX LASSO TYPE PROBLEMS WITH DC PROGRAMMING

Gilles Gasso, Alain Rakotomamonjy and Stéphane Canu

LITIS - EA 4108 - INSA/Universite de Rouen
Avenue de l'Université - 76801 Saint-Etienne du Rouvray Cedex
{gilles.gasso, stephane.canu}@insa-rouen.fr, alain.rakotomamonjy@univ-rouen.fr

ABSTRACT

The paper proposes a novel algorithm for addressing variable selection (or sparsity recovering) problem using non-convex penalties. A generic framework based on a DC programming is presented and yields to an iterative weighted lasso-type problem. We have then showed that many existing approaches for solving such a non-convex problem are particular cases of our algorithm. We also provide some empirical evidence that our algorithm outperforms existing ones.

Keywords: variable selection, non-convex penalization, DC programming, coordinatewise optimization

1. INTRODUCTION

In the context of model estimation, one usually aims at estimate the functional relationship f between some input examples $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and the corresponding outputs $y_1, \cdots, y_n \in \mathbb{R}$. When the dimensionality d is large, it is frequent that a relatively small number of variables are useful to accurately predict the outputs, hence one aims at finding the most predictive variables. One approach is to learn a model for all subsets of variables and then select the best predictive subset. However, this method is solely tractable when d is small. Another approach is to learn the model parameters while simultaneously select variables. Considering a linear model $f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta}$ and a least-squares criterion to assess the model accuracy, the standard approach consists in solving the penalized likelihood to find

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_0} \qquad (1)$$

where $\|\boldsymbol{\beta}\|_{\ell_0}$ is the number of non-zero components in $\boldsymbol{\beta}$. In Problem (1), the complexity of the model is related to the number of involved variables. However, solving (1) is NP-hard and hardly tractable. Hence, to overcome such problem, several works [1, **?**] relax the problem and instead consider this convex problem known as the lasso [1]:

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_1} \qquad (2)$$

The success story of the lasso comes from the fact that the problem can be easily solved either by quadratic programming approach [1], homotopy approaches [2] or coordinate wise optimization [3]. Furthermore, some theoretical arguments state that under certain conditions variable selection with the lasso can be consistent [4].

However, Fan and Li [5] provided some arguments against the lasso: they have shown that the ℓ_1 penalty tends to produce biased estimates for large coefficients. They suggest to replace the ℓ_1 penalty by other penalties that lead to sparse and unbiased model. For this purpose, they advocate that the penalties should be singular at the origin and non-convex like the so-called smoothly clipped absolute deviation (Scad). Penalties like Scad provide some interesting theoretical properties at the expense of introducing more challenges in the optimization problem. For instance, Fan and Li [5] proposed to locally approximate the non-convex penalty with a quadratic function and then use a single Newton step to compute $\boldsymbol{\beta}$. In the same flavor, Zou and Li [6] use a local linear approximation of the penalty function leading to a one-step lasso estimator. From an initial least squares solution, the streamline of these approaches is to solve iteratively a weighted lasso problem where the weights are derived from the previous solution.

In this paper, we propose a general view and framework for solving the non-convex problem resulting from the use of non-convex penalties. Our approach relies on the Difference of Convex (DC) functions programming [7]. Such algorithm involves the decomposition of the non-convex objective function as the difference of two convex functions and solves iteratively a convex problem until a guaranteed convergence to a minimum of the overall optimization problem. We then show that the mentioned methods are particular cases of the proposed methodology. Empiricial experiments shows the benefits of using non-convex penalties instead of the ℓ_1 one and the benefits of using more than two iterations for solving such a non-convex problem.

2. PROBLEM FORMULATION

Assume a set of data $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1,\cdots,n}\}$ where the point $\mathbf{x}_i$ is a d-dimensional predictor and y_i is the associated target output. Without loss of generality, the aim is to predict the value of y_i using a linear functional relationship $f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta}$. The vector of parameters $\boldsymbol{\beta}$ is determined by optimizing the empirical risk $\mathsf{R}^{\text{emp}}[f] = \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i))$ where ℓ is the loss function. To gain some nice properties (for example smoothness or sparsity) of f, a penalty on the parameters is usually considered: $P(\boldsymbol{\beta}) = \sum_{j=1}^{d} \mathsf{g}_j(\beta_j)$ where g_j is a non-negative penalty function that applies to each single coefficient β_j. To easy the presentation, we will consider in the sequel a quadratic loss and the same penalty function for all the coefficients that is $\mathsf{g}_j(\beta_j) = \mathsf{g}(\beta_j), \forall j = 1, \cdots, d$. The optimization problem is therefore

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{j=1}^{d} \mathsf{g}_\lambda(\beta_j) \qquad (3)$$

where $\mathbf{y}$ is the vector containing the target outputs y_i for $i = 1, \cdots, n$, $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the matrix of the covariates and $\mathsf{g}_\lambda(\beta) = \lambda\,\mathsf{g}(\beta)$ with $\lambda \geq 0$ the classical regularization parameter which balances the risk and the penalty.

The most popular penalty function is the ℓ_2 norm which leads to the ridge regression algorithm. But it is well-known that this algorithm does not achieve the sparsity of the obtained model (in fact, in the general case of a ℓ_q penalty $\mathsf{g}_\lambda(\beta) = \lambda|\beta|^q$ with $q > 1$, the solution is not sparse). As we are interested by variable selection and sparsity, other kinds of penalty has to be considered. The lasso algorithm introduced by Tibshirani [1] relies on the ℓ_1 penalty. Based on the convexity of the optimization problem, many approaches were proposed in the literature [2, 8, 3]. Although the consistency of the lasso was proved [4], some experimental results show the bias of the estimated parameters especially for large parameters. Fan and Li [5] have suggested some desired good properties of the penalization term: sparsity, continuity and unbiasness of the estimated parameters. They have proposed the Scad penalty which fulfills these conditions. Contrary to the lasso, this penalty is non-convex and thus requires more optimization efforts.

To ensure more sparsity, other popular concave penalty functions are the ℓ_q penalty with $0 < q < 1$ and the logarithm penalty. The latter function was used in the past as an approximation of the ℓ_0 penalty. Finally, we consider a penalty named in the sequel Zhang's penalty. It corresponds to a two-stage ℓ_1 penalization where in the second stage, the large parameters are not shrinked [9]. Table 1 gives an overview of these penalties. For the log penalty, we consider the logarithm of $|\beta_j|$ shifted by a small quantity $0 < \epsilon \ll 1$ to avoid infinite value when the parameter vanishes. The constant term $-\lambda\log(\varepsilon)$ ensures that the penalty function is non-negative and avoids the trivial solution $\boldsymbol{\beta} = 0$ to (3).

Table 1. Some examples of usual penalty functions

Penalty	Formula
Lasso	$\mathsf{g}_\lambda(\beta_j) = \lambda\|\beta_j\|$
SCAD	$\mathsf{g}_\lambda(\beta_j) = \begin{cases} \lambda\|\beta_j\| & \|\beta_j\| \leq \lambda \\ \frac{-\|\beta_j\|^2 + 2a\lambda\|\beta_j\| - \lambda^2}{2(a-1)} & \lambda < \|\beta_j\| \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & \|\beta_j\| > a\lambda \end{cases}$
Zhang	$\mathsf{g}(\beta_j) = \begin{cases} \lambda\|\beta_j\| & \text{if } \|\beta_j\| < \eta \\ \lambda\eta & \text{otherwise} \end{cases}$
ℓ_q	$\mathsf{g}_\lambda(\beta_j) = \lambda\|\beta_j\|^q \quad \text{with} \quad 0 < q < 1$
Log penalty	$\mathsf{g}_\lambda(\beta_j) = \lambda\log(\|\beta_j\| + \varepsilon) - \lambda\log(\varepsilon)$

Figure 1 illustrates the non-convex penalties of Table 1. We see that the log penalty exhibits a heavy barrier behavior near 0, hence pushing the small parameters toward 0. At a lower lever, the same remark can be made for the ℓ_q penalty.

3. SOLVING THE OPTIMIZATION PROBLEM

3.1. Difference of Convex (DC) penalties Algorithm

Due to the non-convexity of the function g, solving Problem (3) is not an easy task. As mentioned earlier, some authors address this problem using a quadratic local approximation [5] or a one-step lasso scheme [6] beginning form the least-squares solution. Such a procedure can be viewed as an early stopping since they do not iterate the approximation until convergence. We propose rather to address directly the problem using an iterative procedure which converges to a minimum of (3). The algorithm is based on the DC programming which is a robust method consisting in optimizing iteratively a primal-dual subgradient problem [7]. The DC method can address non-convex and non-smooth objective criterion. Applied to our problem, it requires the decomposition of g as the difference of two convex functions:

$$\mathsf{g}_\lambda(\beta_j) = \mathsf{g}_{\text{vex}}(\beta_j) - h(\beta_j) \qquad (4)$$

and leads to the Algorithm 1. At each iteration the convex function h is approximated by its affine maximization. This approximation is its tangent at the current solution leading to a convex optimization problem. Because of the non-convexity of (3), the algorithm cannot always find a global minimum. However, the convergence to a local minimum is theoretically guaranteed (see [7] for more details).

To obtain a lasso-type algorithm, we consider in the sequel, the function $\mathsf{g}_{\text{vex}}(\beta_j) = \lambda|\beta_j|$. From this definition

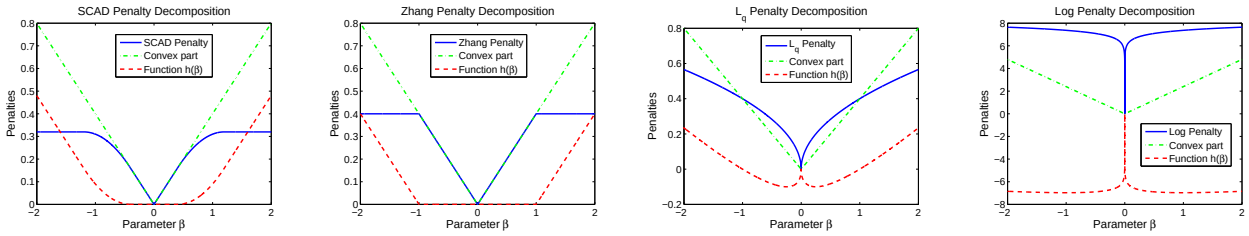

Fig. 1. Illustration of the penalty functions and their proposed DC decomposition. From the left to the right, SCAD penalty, Zhang penalty, ℓ_q and the log-penalty. For clarity sake, the plotted convex part of the log penalty is $6\lambda|\beta|$.

Algorithm 1 Iterative scheme of the DC programming

Set an initial estimation $\boldsymbol{\beta}^0$

repeat

Determine $\boldsymbol{\beta}^{t+1}$ solution of

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{j=1}^{d} \mathrm{g}_{\mathrm{vex}}(\beta_j) - \beta_j h'(\beta_j^t)$$

until convergence of $\boldsymbol{\beta}$

and Equation (4), we derive in Table 2 the expressions of h for the non-convex penalties of Table 1.

Table 2. Expressions of the function h of the DC decomposition for the penalties of Table 1.

Penalty	Expression of h
SCAD	$h_{\mathrm{Scad}}(\beta_j) = \begin{cases} 0 & \text{if} \quad \lvert\beta_j\rvert \le \lambda \\ \frac{\lvert\beta_j\rvert^2 - 2\lambda\lvert\beta_j\rvert + \lambda^2}{2(a-1)}, & \lambda < \lvert\beta_j\rvert \le a\lambda \\ \lambda\lvert\beta_j\rvert - \frac{(a+1)\lambda^2}{2}, & \lvert\beta_j\rvert > a\lambda \end{cases}$
Zhang	$h_{\mathrm{Zhang}}(\beta_j) = \begin{cases} 0 & \text{if} \quad \lvert\beta_j\rvert < \eta \\ \lambda\lvert\beta_j\rvert - \lambda\eta & \text{otherwise} \end{cases}$
ℓ_q	$h_{\ell_q}(\beta_j) = \lambda\left(\lvert\beta_j\rvert - \lvert\beta_j\rvert^q\right)$
Log	$h_{\log}(\beta_j) = \lambda\left(\lvert\beta_j\rvert - \log(\lvert\beta_j\rvert + \varepsilon) + \log\varepsilon\right)$

Graphical illustrations of DC decomposition are depicted in Figure 1. Although the function h for the log and ℓ_q penalties is concave on this figure, we want to emphasis that the penalties operate only on positive values β_j. Hence, only the positive orthant has to be considered in the analysis. In this orthant all the functions h are convex.

3.2. Solving each DC algorithm iteration

The DC decomposition of a penalty is not unique. However, using the proposed decomposition, each iteration of Algorithm 1 solves a lasso type problem. Hence, we can build our algorithm on top of ℓ_1 minimization algorithms and take advantage of all the literature about such a subject. Indeed, at each iteration the convex optimization problem becomes

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{j=1}^{d} \lambda|\beta_j| - \alpha_j\beta_j \tag{5}$$

with $\alpha_j \in \partial h(\beta_j^t)$ where $\partial h(\beta_j^t)$ represents the subdifferential if h is not differentiable. At this step, let give some remarks and insights concerning the subgradient α_j. Firstly, we consider the Scad penalty. As illustrated on Figure 1, h_{Scad} is differentiable and the derivative is given by

$$h'_{\mathrm{Scad}}(\beta_j) = \begin{cases} 0 & \text{if} \quad \lvert\beta_j\rvert \le \lambda \\ \frac{2\beta_j - (2a-1)\lambda\,\mathrm{sign}(\beta_j)}{2(a-1)} & \text{if} \quad \lambda < \lvert\beta_j\rvert \le a\lambda \\ \lambda\,\mathrm{sign}(\beta_j) & \text{if} \quad \lvert\beta_j\rvert > a\lambda \end{cases}$$

Differentiability does not hold for the others penalties. For instance h_{Zhang} is not differentiable at η. The derivative is not unique and we set arbitrary the subgradient at this point to $\lambda\,\mathrm{sign}(\beta_j)$ (one of its possible value). This leads to the following expression:

$$h'_{\mathrm{Zhang}}(\beta_j) = \begin{cases} 0 & \text{if} \quad \lvert\beta_j\rvert < \eta \\ \lambda\,\mathrm{sign}(\beta_j) & \text{otherwise} \end{cases}$$

The function $h_{\log}$ also is not differentiable at 0 because of the non-differentiability of the absolute value function. Using previous arguments, the subgradient is considered null at $\beta_j = 0$ and $h'_{\log}(\beta_j) = \lambda\mathrm{sign}(\beta_j)\left(1 - \frac{1}{|\beta_j|+\varepsilon}\right)$ if $\beta_j \neq 0$. We can see that this derivative doesn't blow-up if the parameter is close to the null value. Finally the subgradient for the ℓ_q penalty is derived similarly: it is null at 0 and equal to $h'_{\ell_q}(\beta_j) = \lambda\mathrm{sign}(\beta_j)\left(1 - \frac{q}{|\beta_j|^{1-q}}\right)$ if $\beta_j \neq 0$. Notice that the DC algorithm still applies when using these subgradients [7].

These elements of the algorithm being clarified, we focus on the optimization problem (5). Defining each parameter as the difference of two positive terms $\beta_j = \beta_j^+ - \beta_j^-$,

the latter problem can be turned into:

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \left\| \mathbf{y} - \mathbf{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) \right\|^2 + \sum_{j=1}^{d} \lambda(\beta_j^+ + \beta_j^-) - \sum_{j=1}^{d} \alpha_j(\beta_j^+ - \beta_j^-)$$
$$\text{s.t.} \qquad \beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \quad \forall j = 1, \cdots, d$$

where $\boldsymbol{\beta}^+ \in \mathbb{R}^d$ (respectively $\boldsymbol{\beta}^-$) is the vector containing the parameters β_j^+ (respectively β_j^-). The details of the algorithm we developed are the following. At first, let us rewrite the problem in a more general fashion :

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \left\| \mathbf{y} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} \right\|^2 + \sum_{j=1}^{2d} \lambda_j \tilde{\beta}_j \qquad (6)$$
$$\text{s.t.} \qquad \tilde{\beta}_j \geq 0 \quad \forall j = 1, \cdots, 2d$$

with $\tilde{\mathbf{X}} = [\mathbf{X} \ -\mathbf{X}]$, $\tilde{\boldsymbol{\beta}} = \begin{bmatrix} \boldsymbol{\beta}^+ \\ \boldsymbol{\beta}^- \end{bmatrix}$, $\lambda_j = \lambda - \alpha_j$ for $j = 1, \cdots, d$ and $\lambda_j = \lambda + \alpha_j$ for $j = d+1, \cdots, 2d$. Writing the Lagrangian function of this constrained optimization problem and setting its derivatives to zero gives the following optimality conditions

$$\tilde{\mathrm{x}}_j^{\mathsf{T}}(\mathbf{y} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}) - \lambda_j = 0 \quad \text{if} \quad \tilde{\beta}_j > 0$$
$$\tilde{\mathrm{x}}_j^{\mathsf{T}}(\mathbf{y} - \tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}}) < \lambda_j \quad \text{if} \quad \tilde{\beta}_j = 0$$

where $\tilde{\mathrm{x}}_j$ represents the j^{th} column of the matrix $\tilde{\mathbf{X}}$. The problem (6) is strongly related to the lasso problem. Indeed, it is a modified version of the lasso with positive constraints on the variables. Thus, solving problem (6) can be done by building upon algorithms solving the lasso.

The algorithm we use is based on ideas drawing from coordinatewise optimization and active constraints based approaches. Recently, [3] has shown that the coordinatewise optimization algorithm is an efficient and competitive approach compared to homotopy or quadratic programming approaches [8, 2, 1]. Furthermore, taking into account the optimality constraints for selecting the variables to optimize makes the algorithm still more efficient (see [10]).

The idea is to optimize (6) according to a single variable $\tilde{\beta}_j$ at a time. The procedure is to select a coefficient that violates optimality conditions and to optimize a subproblem associated to that variable while keeping all other variables fixed. This step is then repeated as long as there exists a variable that violates its constraint. This method is similar to the active constraint method used in SimpleSVM [11]. The overall algorithm for coordinatewise optimization is given in Algo 2. We have denoted as I_0 the set of coefficients $\tilde{\beta}$ that vanish and I_P the set of positive coefficients $\tilde{\beta}$.

Optimizing each single variable can be done in several ways. One can use a Newton method as proposed by [12] or a coordinatewise update method as described by [3]. After having compared both approaches, we have figured out that the Newton method is faster than the coordinatewise update. Hence, we consider only that method.

Algorithm 2 Coordinatewise descent optimization

```
Suppose given initial estimation β̃_init
while a contraint for I_0 are violated do
  Find a violator β̃_x in I_0
  repeat
    Optimize with respect to β̃_x
    Find a violator β̃_x in I_P
  until the set of violator in I_P is empty
end while
```

Note that another advantage of our algorithm is that it can be initialized with any coefficient vector $\boldsymbol{\beta}$. According to the DC algorithm procedure, this coordinatewise algorithm is performed several times until $\boldsymbol{\beta}$ reaches a fixed point. At each iteration, the algorithm is initialized with the previous solution. Since, we expect that from iteration to iteration the parameters change slightly, by doing so, we expect that the cost of each DC algorithm iteration is low.

3.3. Relations with other algorithms

Recently, a lot of works dealt with the lasso problem. Most of them have proved that the lasso in some situations does not insure the consistency, and thus they have introduced alternative lasso problems. For instance, Zhang has proposed a two-stage lasso [9], Zou has introduced the adaptive lasso [13], while Zou and Li improved the Scad by using Local Linear Approximation (LLA) [6]. All these approaches can be considered as particular cases of our general approach.

Indeed, if we use the penalty function of the 2-stage lasso of Zhang in our algorithm, and by initializing the algorithm with a coefficient vector $\boldsymbol{\beta} = \mathbf{0}$, then after two iterations, we exactly recover the 2-stage lasso. However, in our case, more iterations can occur until the stopping condition is met. Hence, our algorithm insures a better convergence of the non-convex optimization problem towards a minimum. Recently, Zou and Li advocate the use of the LLA to address the Scad penalty problem. This is equivalent to one-step of our DC procedure with a least-squares initialization. However, since they only perform a single step, using our DC approach provides a better convergence particularly if the initializing vector is not optimal. The same remark holds for the adaptive lasso which is a single iteration of our algorithm with a log penalty and an initial least-squares solution.

4. NUMERICAL EXPERIMENTS

In this section, we present some experiments that demonstrate the benefits of using non-convex penalties and DC

programming. First, we aim at confirming the evidence, already proved by others, that non-convex penalties are performing better than the ℓ_1 penalty for recovering sparse signal. Then, we also empirically prove that using a DC algorithm and a thus reweighted-like iterative scheme leads to better performance than a one-step estimator such as the adaptive lasso [13] or the one-step Scad [6].

For these purposes, we have designed an experiment where the true coefficient vector $\boldsymbol{\beta}$ is known. For each trial, we have done the following. We have selected a coefficient vector $\boldsymbol{\beta}$ of dimension $d = 256$ for which the number of active variables k is varying. The k non-zeros positions are chosen randomly and the non-zeros values are either drawn from zero-mean unit variance Gaussian distribution or a symmetric Bernouilli distribution with ± 1 values. We have sampled a random matrix $\mathbf{X}$ of dimension $n \times d$ with columns drawn uniformly from the surface of a unit hypersphere. Then, the target vector is obtained as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\xi}$ where $\boldsymbol{\xi}$ is vector drawn from i.i.d Gaussian distribution with zero-mean and variance σ_b^2 determined from a a given Signal-To-Noise as $\sigma_b^2 = \frac{1}{n}\|\mathbf{X}\boldsymbol{\beta}\|^2 \cdot 10^{-SNR/10}$.

For each trial, we have run our algorithm with different non-convex penalties as well as different methods available from the literature. As a performance criterion, we have used the Hamming distance between the support of the true coefficient vector and the estimated one with the support of $\boldsymbol{\beta}$ defined as $\text{supp}(\boldsymbol{\beta}) = \{j : \beta_j > \epsilon\}$ where ϵ is a threshold that allows us to neglect some true non-zero coefficients that may be obliterate by the noise.

Note also that all the algorithms we used share the same structure (penalized least-squares) with the same hyperparameter. Thus, the results we reported here do not consider the problem of selecting the hyperparameter λ. This make algorithms comparison easier since difference in performances can not be attributed to poor model selection (which in any case can be performed by using the same method and criterion).

Figure 2 shows the variation of performance with respect to regularization parameter λ of the lasso estimate and the estimates obtained with our algorithm associated with different non-convex penalties. From these plots, we can see that using non-convex penalties allow better recovery of the sparsity support of underlying signals regardless to the different experimental settings (low $SNR = 10$ or high $SNR = 30$ signal to noise ratio or kind of entries Gaussian or Bernouilli). An interesting point that can be noted is that using non-convex penalties are less valuable when the coefficient vector entries are ± 1. We can think that since these entries take uniformly large values, the sparsity support becomes easy to estimate, so that the lasso can produce a correct estimate.

Table 3 and Figure 3 summarize gains in performance that are achieved by our algorithm using different penalties compared to their one-step counterparts. Table 3 shows the minimal average Hamming error obtained when varying λ and the average number of iterations needed before convergence. Note that since the adaptive lasso, one-step Scad, Zhang's method and one-step ℓ_q can all be viewed as a one-step reweighted lasso, they all need 2 iterations. From the table, we can see that for the experimental settings we use (k=50 and SNR = 30 dB), multiple iterations of DC algorithms always leads to better performance. Such empirical evidence makes clear the necessity of an optimization algorithm that is guaranteed to converge toward a local minimum. Figure 3 depicts the error ratio (full iterations/ two iterations) with respect to the hyperparameter λ. Interestingly, we see that the log and the ℓ_q penalties are the two ones that benefit the more from the full convergence of the DC algorithm whereas, Zhang's penalty shows only little performance improvement. This figure also reveals the importance of hyperparameter selection. Indeed, we can see that poor model selection can lead to performance degradation of our proposed algorithm. However, such a situation occurs only for high value of λ.

Table 3. Summary of performances. k=50, SNR=30 dB. Top) Gaussian coefficients Down) Bernouilli coefficients

Algorithms	min. error	Nb. Iter
Lasso	0.14 ± 0.02	1
Log	0.02 ± 0.01	9.4
Adapt. Lasso	0.03 ± 0.01	2
SCAD	0.02 ± 0.01	8.50
1-SCAD	0.05 ± 0.03	2
Zhang	0.04 ± 0.01	4.0
1-Zhang	0.05 ± 0.02	2
ℓ_q	0.02 ± 0.01	4.4
1-ℓ_q	0.04 ± 0.1	2

Algorithms	min. error	Nb. Iter
Lasso	0.15 ± 0.02	1
Log	0.08 ± 0.06	9.43
Adapt. Lasso	0.10 ± 0.07	2
SCAD	0.11 ± 0.10	6.90
1-SCAD	0.12 ± 0.09	2
Zhang	0.12 ± 0.07	4.93
1-Zhang	0.14 ± 0.04	2
ℓ_q	0.09 ± 0.08	7.2
1-ℓ_q	0.12 ± 0.10	2

5. CONCLUSIONS

The paper has addressed the variable selection problem with non-convex penalties. Our algorithm is an iterative weighted

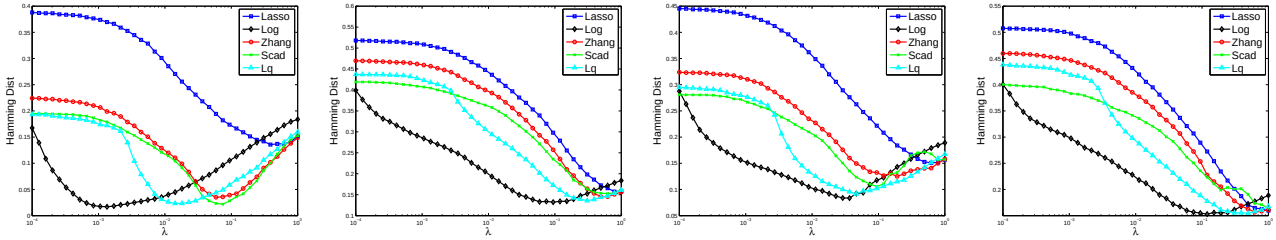

Fig. 2. The average (over 30 trials) Hamming distance of different algorithms with respect to the trade-off hyperparameter λ and for a number of non-zero elements $k = 50$. From left to right, the two first figures plot the performance for Gaussian entries coefficient vector while the two last graphics correspond to Bernouilli entries. Graphics 1 and 3 depict performances for a high SNR (30 dB) and the others for a low SNR (10 dB).

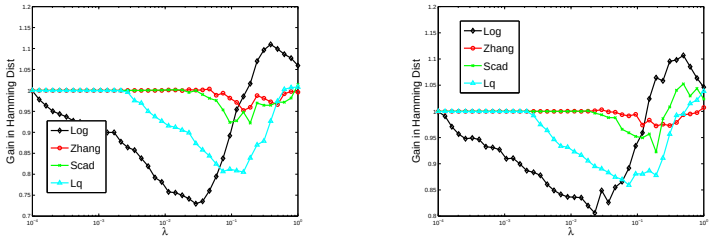

Fig. 3. Average Gain in performance with respect to λ when comparing the estimate produced by our DC algorithm after two iterations and after full convergence of the algorithm. Here we have $k = 50$ and SNR =30 dB. Left) Gaussian coefficient vectors Right) Bernouilli coefficient vectors.

lasso type optimization, thus taking advantage of all the literature on the lasso. The convergence towards a minimum of the solved problem is guaranteed. The simulation results exhibit an improvement of performances compared to existing procedures. However, more experiences and applications have to be carried. Other perspectives are the optimal tuning of the hyperparameters involved in the problem.

6. REFERENCES

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 46, pp. 431–439, 1996.

[2] M.R. Osborne, B. Presnell, and B.A. Turlach, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 319–337, 2000.

[3] J. Friedman, T. Hastie, H. Hfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[4] P. Zhao and B. Yu, "On model selection consistency of lasso," *Journal of Machine Learning Research*, vol. 7, pp. 2541–2563, 2006.

[5] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[6] H. Zou and R. Li, "One-step sparse estimates in nonconcave penalized likelihood models," *The Annals of Statistics*, To appear.

[7] P. D. Tao and L. T. Hoai An, "Dc optimization algorithms for solving the trust region subproblem," *SIAM Journal of Optimization*, vol. 8, no. 2, pp. 476–505, 1998.

[8] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[9] T. Zhang, "Some sharp performance bounds for least squares regression with l_1 regularization," Tech. Rep., Dept. of Statistics, Rutgers University, 2007.

[10] "S. Shevade and S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[11] S. V. N. Vishwanathan, A. J. Smola, and M. Murty, "SimpleSVM," in *International Conference on Machine Learning*, 2003.

[12] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text," *Technometrics*, vol. 49, pp. 291–304, 2007.

[13] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006.