Journal of Statistical Software

August 2014, Volume 59, Issue 8. http://www.jstatsoft.org/

dglars: An R Package to Estimate Sparse Generalized Linear Models

Luigi Augugliaro
University of Palermo

Angelo M. Mineo
University of Palermo

Ernst C. Wit
University of Groningen

Abstract

dglars is a publicly available R package that implements the method proposed in Augugliaro, Mineo, and Wit (2013), developed to study the sparse structure of a generalized linear model. This method, called dgLARS, is based on a differential geometrical extension of the least angle regression method proposed in Efron, Hastie, Johnstone, and Tibshirani (2004). The core of the **dglars** package consists of two algorithms implemented in Fortran 90 to efficiently compute the solution curve: a predictor-corrector algorithm, proposed in Augugliaro *et al.* (2013), and a cyclic coordinate descent algorithm, proposed in Augugliaro, Mineo, and Wit (2012). The latter algorithm, as shown here, is significantly faster than the predictor-corrector algorithm. For comparison purposes, we have implemented both algorithms.

Keywords: differential geometry, generalized linear models, dgLARS, predictor-corrector algorithm, cyclic coordinate descent algorithm, sparse models, variable selection.

1. Introduction

Nowadays, high-dimensional datasets, in which the number of predictors, say p, is larger than the sample size N, are becoming more and more common. Routine monitoring, e.g., of web traffic or satellite surveyance, and high-throughput screening methods in biology and chemistry, raise the tantalizing prospect of being able to explain and predict a wide range of phenomena. What makes high-dimensional statistical inference possible is the assumption that the regression function lies in a low-dimensional manifold (Fan and Lv 2010). In a regression setting, this means that the coefficient parameter vector has a sparse structure. Modern statistical methods developed to cope with this problem are usually based on the idea of using a penalty function to estimate a sparse solution curve embedded in the parameter space and then to find the point that represents the best compromise between sparsity and fit of the model. Recent statistical literature has a great number of contributions devoted to

this problem: some important examples are the L_1-penalty method, originally proposed in Tibshirani (1996), the SCAD method (Fan and Li 2001), the Dantzig selector (Candes and Tao 2007), which was extended to generalized linear models (GLMs) in James and Radchenko (2009), and the MC+ penalty function introduced in Zhang (2010), among others.

Differently from the methods cited above, Augugliaro *et al.* (2013) propose a new approach based on the differential geometrical representation of a GLM. The method does not require an explicit penalty function, because it introduces sparsity more directly: it defines the sparsest continuous solution path with a pointwise maximum likelihood estimate (MLE). It has been called differential geometric LARS (dgLARS) because it generalizes the geometrical ideas underlying least angle regression (LARS) proposed in Efron *et al.* (2004). As emphasized in Augugliaro *et al.* (2013), LARS is not only "an important contribution to statistical computing", as suggested in Madigan and Ridgeway (2004), but is a proper likelihood method in its own right: it can be generalized to any model and its success does not depend on the arbitrary match of the constraint and the objective function, as is the case in penalized inference methods. In particular, using the differential geometric characterization of the classical signed Rao's score test statistic, dgLARS gains important theoretical properties that are not shared by other methods.

From a computational point of view, the dgLARS method consists essentially in computing the implicitly defined solution curve. In Augugliaro *et al.* (2013) this problem is satisfactorily solved by using a predictor-corrector (PC) algorithm, that however has the drawback of becoming intractable when working with thousands of predictors, since in the predictor step of this algorithm the number of arithmetic operations scales as the cube of the number of predictors. To overcome this problem, Augugliaro *et al.* (2012) propose a much more efficient cyclic coordinate descend (CCD) algorithm, which connects the original dgLARS problem with an iterative reweighted least squares (IRLS) algorithm.

In this paper we present the R (R Core Team 2014) **dglars** package (Augugliaro 2014), that implements both the algorithms to compute the solution curve implicitly defined by dgLARS. In this version of the package only the Poisson and binomial family are implemented, other families will be added in future versions of the package. The implemented functions return an object inheriting from an S3 class for which suitable methods have been implemented. The package **dglars** is available under the general public license (GPL ≥ 2) from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=dglars`.

The paper is organized as follows. In Section 2 we give some methodological background of the proposed method. More specifically, in Section 2.1 we introduce the dgLARS method by giving some essential clues to the theory underlying a GLM from a differential geometric point of view. In Section 2.2 we give a description of the implemented algorithms and in Section 2.3 we describe the notion of generalized degrees of freedom. Section 3 is devoted to the description of the main functions implemented in the **dglars** package. In Section 4 we compare the behavior of the proposed method with the L_1-penalized MLE by simulation studies and in Section 5 we use the functions implemented in our package to study two real data sets. Finally, in Section 6 we draw some conclusions.

2. Methodological background

In this section we give a rough overview of the method. The reader interested in more of

the differential geometric details of this method is referred to Augugliaro *et al.* (2013). The dgLARS method defines a continuous solution path for a GLM, with on one extreme of the path the MLE and on the other extreme the intercept-only estimate. The aim of the method is to define the most efficient model – in likelihood terms – that uses the fewest variables. In order to describe the method, we introduce the GLM and its dgLARS solution path in Section 2.1. The computational aspects are given in Section 2.2.

2.1. Description of the dgLARS method

Let $\mathcal{Y}$ be a scalar random variable with probability density function belonging to the exponential family, i.e.,

$$p(y; \theta_i, \phi) = \exp\{(y\theta - b(\theta))/a(\phi) + c(y, \phi)\},$$

where $\theta \in \Theta \subseteq \mathbb{R}$ is the canonical parameter, $\phi \in \Phi \subseteq \mathbb{R}^+$ the dispersion parameter and $a(\cdot), b(\cdot)$ and $c(\cdot, \cdot)$ are specific given functions. The expected value of $\mathcal{Y}$ is related to the canonical parameter by the mean value mapping, namely

$$\mathsf{E}(\mathcal{Y}) = \mu = \partial b(\theta)/\partial\theta,$$

similarly, the variance of $\mathcal{Y}$ is related to its expected value by the identity $\mathsf{VAR}(\mathcal{Y}) = a(\phi)V(\mu)$, where $V(\mu)$ is called variance function. In the following we shall assume that the dispersion parameter is fixed, in this case, without loss of generality we can set $\phi = 1$. Let $\boldsymbol{\mathcal{X}}$ be the p-dimensional vector of random predictors. A GLM is based on the assumption that the conditional expected value of $\mathcal{Y}$ given $\boldsymbol{\mathcal{X}} = \mathbf{x}$ is specified by the link function $h(\cdot)$, namely

$$h(\mathsf{E}(\mathcal{Y}|\boldsymbol{\mathcal{X}} = \mathbf{x})) = \beta_0 + \sum_{m=1}^{p} x_m\beta_m.$$

This means that the probability density function of $\mathcal{Y}$ is defined with respect to the mean value $\mu(\boldsymbol{\beta}) = h^{-1}(\mathbf{x}^\top\boldsymbol{\beta})$, where $x_0 = 1$. Under the assumption that we are working with N independent copies of the pair $(\mathcal{Y}, \boldsymbol{\mathcal{X}})$ the conditional probability density function can be written as $p(\mathbf{y}; \boldsymbol{\mu}(\boldsymbol{\beta})) = \prod_{i=1}^{N} p(y_i; \mu_i(\boldsymbol{\beta}))$. In the following of this paper we shall use $\ell(\boldsymbol{\beta}; \mathbf{y}) = \log p(\mathbf{y}; \boldsymbol{\mu}(\boldsymbol{\beta}))$ as notation for the conditional log-likelihood function.

The mth score function, measuring the likelihood increase in the mth direction is given as

$$\partial_m\ell(\boldsymbol{\beta}; \boldsymbol{y}) = \partial \log p(\boldsymbol{y}; \boldsymbol{\mu}(\boldsymbol{\beta}))/\partial\beta_m.$$

To measure the scale of this change, relative to the size of the mth predictor, we scale the score function by the square root of the conditional Fisher information, i.e.,

$$I_m(\boldsymbol{\beta}) = \mathsf{E}(\partial_m\ell(\boldsymbol{\beta}; \boldsymbol{\mathcal{Y}})^2),$$

to obtain the conditional Rao's score statistic,

$$r_m^u(\boldsymbol{\beta}) = \frac{\partial_m\ell(\boldsymbol{\beta}; \mathbf{y})}{\sqrt{I_m(\boldsymbol{\beta})}}.$$

At the MLE $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$, the Rao's score statistics are equal to zero, since the derivatives of the likelihood are zero. On the other hand, for the minimal model with only the MLE of the

intercept $\hat{\boldsymbol{\beta}}_0 = (\hat{\beta}_0, 0, \ldots, 0)$, we define $\gamma_{\max}$ to be the largest absolute value of the Rao's score statistic at $\hat{\boldsymbol{\beta}}_0$, i.e.,

$$\gamma_{\max} = \max_{m=1,\ldots,p} |r^u_m(\hat{\boldsymbol{\beta}}_0)|.$$

This value points out the best instantaneous "normalized" contribution to the likelihood of a single variable. This variable would make an excellent candidate for being included in the model. With these definitions in hand, we can now define the dgLARS estimator.

Definition 1 *Let $\gamma \in [0, \gamma_{\max}]$ be a fixed value. The dgLARS estimator of a given GLM, denoted by $\hat{\boldsymbol{\beta}}(\gamma) \in \mathbb{R}^p$, is such that the following conditions are satisfied:*

$$|r^u_m(\hat{\boldsymbol{\beta}}(\gamma))| = \gamma, \qquad \forall\, m \in \mathcal{A}(\gamma), \tag{1}$$
$$|r^u_m(\hat{\boldsymbol{\beta}}(\gamma))| < \gamma, \qquad \forall\, m \notin \mathcal{A}(\gamma). \tag{2}$$

where $\mathcal{A}(\gamma) = \{m : \hat{\beta}_m(\gamma) \neq 0\}$ is called active set.

Here we have defined the dgLARS estimator in terms of the Rao's score statistic. Augugliaro *et al.* (2013) show that Definition 1 follows naturally from a differential geometric interpretation of a GLM that allows us to generalize the LARS method introduced in Efron *et al.* (2004). The dgLARS method generalizes the geometric description of LARS and is based on the following simple differential geometric identity

$$r^u_m(\boldsymbol{\beta}) = \cos \rho_m(\boldsymbol{\beta}) \cdot \|r_{\boldsymbol{\beta}}(\boldsymbol{\mathcal{Y}})\|_{p(\boldsymbol{\mu}(\beta))},$$

where $\rho_m(\boldsymbol{\beta})$ is the angle between $\partial_m \ell(\boldsymbol{\beta}; \boldsymbol{\mathcal{Y}})$ and the tangent residual vector $r_{\boldsymbol{\beta}}(\boldsymbol{\mathcal{Y}})$ while $\|r_{\boldsymbol{\beta}}(\boldsymbol{\mathcal{Y}})\|_{p(\boldsymbol{\mu}(\beta))}$ is the length of this vector – which, crucially, does not depend on variable m. Using Figure 1 the dgLARS method can be described in the following way. First the method selects the predictor, say $\mathcal{X}_{a_1}$, whose basis vector $\partial_{a_1} \ell(\hat{\boldsymbol{\beta}}(\gamma_{\max}); \boldsymbol{\mathcal{Y}})$ has the smallest angle with the tangent residual vector, and includes it in the active set $\mathcal{A}(\gamma^{(1)}) = \{a_0, a_1\}$, where a_0 stands for the intercept and $\gamma^{(1)} = \gamma_{\max}$. The solution curve $\hat{\boldsymbol{\beta}}(\gamma) = (\hat{\beta}_{a_0}(\gamma), \hat{\beta}_{a_1}(\gamma), 0, \ldots, 0)^\top$ is chosen in such a way that the tangent residual vector is always orthogonal to the basis $\partial_{a_0} \ell(\hat{\boldsymbol{\beta}}(\gamma); \boldsymbol{\mathcal{Y}})$, while the direction of the curve $\hat{\boldsymbol{\beta}}(\gamma)$ is defined by the projection of the tangent residual vector onto the basis vector $\partial_{a_1} \ell(\hat{\boldsymbol{\beta}}(\gamma); \boldsymbol{\mathcal{Y}})$. The curve $\hat{\boldsymbol{\beta}}(\gamma)$ continues as defined above until $\gamma = \gamma^{(2)}$, for which there exists a new predictor, say $\mathcal{X}_{a_2}$, that satisfies the equiangularity condition, namely

$$\rho_{a_1}(\hat{\boldsymbol{\beta}}(\gamma^{(2)})) = \rho_{a_2}(\hat{\boldsymbol{\beta}}(\gamma^{(2)})). \tag{3}$$

At this point $\mathcal{X}_{a_2}$ is included in $\mathcal{A}(\gamma^{(2)})$ and the curve $\hat{\boldsymbol{\beta}}(\gamma) = (\beta_{a_0}(\gamma), \beta_{a_1}(\gamma), \beta_{a_2}(\gamma), 0, \ldots, 0)^\top$ continues, such that the tangent residual vector is always orthogonal to the basis vector $\partial_{a_0} \ell(\hat{\boldsymbol{\beta}}(\gamma); \boldsymbol{\mathcal{Y}})$ and with direction defined by the tangent vector that bisects the angle between $\partial_{a_1} \ell(\hat{\boldsymbol{\beta}}(\gamma); \boldsymbol{\mathcal{Y}})$ and $\partial_{a_2} \ell(\hat{\boldsymbol{\beta}}(\gamma); \boldsymbol{\mathcal{Y}})$, as shown on the right side of Figure 1.

Efron *et al.* (2004) show that the LASSO solution curve can be obtained by a simple modification of the LARS method. Let $\hat{\boldsymbol{\beta}}(\gamma)$ be the solution of a GLM penalized using the L_1-penalty function, then it is easy to show that the sign of any non-zero coefficient has to agree with the sign of the score function, namely

$$\operatorname{sign}(\partial_m \ell(\hat{\boldsymbol{\beta}}(\gamma); \mathbf{y})) = \operatorname{sign}(\hat{\beta}_m(\gamma)).$$

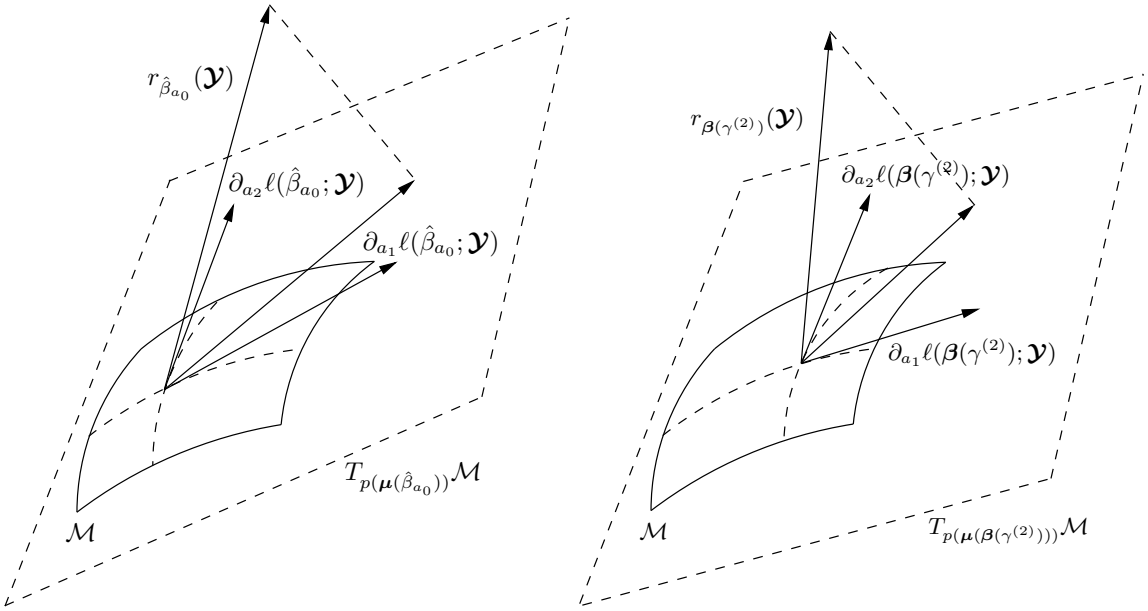

Figure 1: Differential geometrical description of the dgLARS method for a GLM with two covariates; in the left side the first predictor $\mathcal{X}_{a_1}$ is found and included in the active set; in the right side the generalized equiangularity condition (Equation 3) is satisfied for $\mathcal{X}_{a_1}$ and $\mathcal{X}_{a_2}$.

When this condition is violated the corresponding predictor is removed from the active set. Like the original LARS method, also the dgLARS method can be easily modified to compute a differential geometric extension of the LASSO solution curve, called the dgLASSO solution curve.

Definition 2 *Let $\gamma \in [0, \gamma_{\max}]$ be a fixed value. The dgLASSO estimator of a GLM is given by Condition 1 and Condition 2, such that also the following restrictions on the signs of the non-zero coefficients are satisfied*

$$sign(r^u_m(\hat{\boldsymbol{\beta}}(\gamma))) = sign(\hat{\beta}_m(\gamma))), \qquad \forall\, m \in \mathcal{A}(\gamma).$$

The dgLASSO method computes the dgLASSO solution curve in the same way as the dgLARS method, but it removes a predictor from the active set when the sign of the corresponding estimate is not in agreement with the sign of the Rao's score test statistic. Like for the LASSO and SCAD estimator also for dgLARS and dgLASSO estimator the problem of how to estimate the dispersion parameter ϕ is still an open question and theoretical results are not available. More details about the dgLASSO method will be given in the following section.

2.2. Computational aspects

From a computational point of view, the problem of how to estimate the dgLARS solution curve can be formalized in this way: we want to find a decreasing sequence of p values of the tuning parameter γ, say

$$0 \leq \gamma^{(p)} \leq \ldots \leq \gamma^{(2)} \leq \gamma^{(1)}$$

such that for any $\gamma \in (\gamma^{(k+1)}; \gamma^{(k)}]$ the dgLARS estimator satisfies the following conditions:

$$\begin{aligned} \mathcal{A}(\gamma) &= \{a_1, a_2, \ldots, a_k\}, \\ |r^u_{a_i}(\hat{\boldsymbol{\beta}}(\gamma))| &= |r^u_{a_j}(\hat{\boldsymbol{\beta}}(\gamma))|, \quad \forall\, a_i, a_j \in \mathcal{A}(\gamma), \\ |r^u_{a^c_h}(\hat{\boldsymbol{\beta}}(\gamma))| &< |r^u_{a_i}(\hat{\boldsymbol{\beta}}(\gamma))|, \quad \forall\, a^c_h \in \mathcal{A}^c(\gamma) \text{ and } \forall\, a_i \in \mathcal{A}(\gamma), \end{aligned} \tag{4}$$

where $\mathcal{A}^c(\gamma)$ is the complement of the active set. The new predictor is included in the active set at $\gamma = \gamma^{(k+1)}$ where the following condition is satisfied

$$\exists\, a^c_h \in \mathcal{A}^c(\gamma^{(k+1)}) \;:\; |r^u_{a^c_h}(\hat{\boldsymbol{\beta}}(\gamma^{(k+1)}))| = |r^u_{a_i}(\hat{\boldsymbol{\beta}}(\gamma^{(k+1)}))|, \quad \forall a_i \in \mathcal{A}(\gamma^{(k+1)}).$$

When we want to estimate the dgLASSO solution curve it is also necessary to add the following condition

$$\exists\, a_i \in \mathcal{A}(\gamma^{(k+1)}) \;:\; \hat{\beta}_{a_i}(\gamma^{(k+1)}) = 0. \tag{5}$$

Since for any $a_i \in \mathcal{A}(\gamma)$ the sign of the $r^u_{a_i}(\hat{\boldsymbol{\beta}}(\gamma))$ does not change, it is easy to see that Equation 5 identifies a value of the tuning parameter such that for any $\gamma < \gamma^{(k+1)}$ we have that $\text{sign}(r^u_{a_i}(\hat{\boldsymbol{\beta}}(\gamma))) \neq \text{sign}(\hat{\beta}_{a_i}(\gamma))$ which means that the predictor a_i must be removed from the active set.

Augugliaro *et al.* (2013) propose to use a predictor-corrector (PC) method to compute the dgLARS/dgLASSO solution curve. From a computational point of view, the main problem of using the PC algorithm is related to the number of arithmetic operations needed to compute the Euler predictor that is involved in the computation and that requires the inversion of an adequate Jacobian matrix. In the actual implementation of the PC algorithm, the prediction step has complexity $O(|\mathcal{A}(\gamma)|^3)$ but it can be improved to $O(|\mathcal{A}(\gamma)|^{2.376})$ using the Coppersmith-Winograd algorithm. In any case, these results make such an algorithm cumbersome in a high-dimensional setting. To overcome this problem, Augugliaro *et al.* (2012) propose a cyclic coordinate descent (CCD) method, which can be defined relating the original dgLARS problem to a sequence of penalized weighted least squares problems. In a high-dimensional setting, this method is computationally more efficient than the predictor-corrector algorithm to compute the dgLARS solution curve. In the following we describe both algorithms that we have implemented in the **dglars** package.

Predictor-corrector (PC) algorithm

In order to make this paper self-contained, we briefly review the PC algorithm originally proposed to compute the dgLARS/dgLASSO solution curve. For more details the interested reader is referred to Augugliaro *et al.* (2013). The basic idea underlying the PC algorithm (Allgower and Georg 2003) is to trace a curve implicitly defined by a system of non-linear equations. The curve is obtained by generating a sequence of points satisfying a chosen tolerance criterion.

Let $\gamma \in (\gamma^{(k+1)}; \gamma^{(k)}]$ be a fixed value of the tuning parameter, such that k predictors are included in the active set, i.e., $\mathcal{A}(\gamma) = \{a_1, a_2, \ldots, a_k\}$. The corresponding point of the solution curve will be denoted by $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma) = (\hat{\boldsymbol{\beta}}_{a_0}(\gamma), \hat{\boldsymbol{\beta}}_{a_1}(\gamma), \ldots, \hat{\boldsymbol{\beta}}_{a_k}(\gamma))^\top$. Using Equation 4, the solution path satisfies the relationship

$$|r^u_{a_1}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))| = |r^u_{a_2}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))| = \ldots = |r^u_{a_k}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))|,$$

which means that $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ solves the following system of $k+1$ non-linear equations

$$\begin{cases} \partial_{a_0}\ell(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma);\mathbf{y}) &= 0, \\ r^u_{a_1}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)) &= v_{a_1}\cdot\gamma, \\ \vdots & \vdots \\ r^u_{a_k}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)) &= v_{a_k}\cdot\gamma, \end{cases} \tag{6}$$

where $v_{a_i} = \text{sign}(r^u_{a_i}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma^{(k)})))$. In order to simplify our notation, we define $\tilde{\boldsymbol{\varphi}}_{\mathcal{A}}(\gamma) = \boldsymbol{\varphi}_{\mathcal{A}}(\gamma) - \mathbf{v}_{\mathcal{A}}\gamma$, where $\boldsymbol{\varphi}_{\mathcal{A}}(\gamma) = (\partial_{a_0}\ell(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma);\mathbf{y}), r^u_{a_1}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)), \ldots, r^u_{a_k}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)))^\top$ and $\mathbf{v}_{\mathcal{A}} = (0, v_{a_1}, \ldots, v_{a_k})^\top$. By differentiating $\tilde{\boldsymbol{\varphi}}_{\mathcal{A}}(\gamma)$ with respect to γ, we can locally approximate the solution curve at $\gamma - \Delta\gamma$ by the following expression,

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma) \approx \tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma) = \hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma) - \Delta\gamma \cdot \left[\frac{\partial \boldsymbol{\varphi}_{\mathcal{A}}(\gamma)}{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)}\right]^{-1} \mathbf{v}_{\mathcal{A}}, \tag{7}$$

where $\Delta\gamma \in [0; \gamma - \gamma^{(k+1)}]$ and $\partial\boldsymbol{\varphi}_{\mathcal{A}}(\gamma)/\partial\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ is the Jacobian matrix of the vector function $\boldsymbol{\varphi}_{\mathcal{A}}(\gamma)$ evaluated at $\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$. To approximate the point that lies on the dgLARS solution curve, in Augugliaro *et al.* (2013) the step size $\Delta\gamma$ is approximated by the following expression

$$\Delta\gamma^{\text{in}} = \min_{a^c_h \in \mathcal{A}^c(\gamma)}{}^{+} \left\{ \frac{\gamma - r^u_{a^c_h}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{1 - \frac{dr^u_{a^c_h}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{d\gamma}}; \frac{\gamma + r^u_{a^c_h}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{1 + \frac{dr^u_{a^c_h}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))}{d\gamma}} \right\}, \tag{8}$$

where "$\min^+$" indicates that the minimum is taken over only positive components within each choice of a^c_h. Approximation 8 generalizes the step size proposed for the LARS method (Efron *et al.* 2004) and approximates the step size corresponding to the inclusion of a new predictor in the active set. Equation 7 with step size given by Equation 8 is used for the predictor step of the PC algorithm. In the corrector step, $\tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma - \Delta\gamma^{\text{in}})$ is used as starting point for the Newton-Raphson algorithm that is used to solve System 6.

When we want to estimate the dgLASSO solution curve it is necessary to adjust the step size given by Equation 8 in order to consider Equation 5. Using Equation 7, it is easy to see that the first sign change will, approximately, occur at

$$\Delta\gamma^{\text{out}} = \min_{a_i \in \mathcal{A}}\{\beta_{a_i}(\gamma^{(k)})/d_{a_i}(\gamma^{(k)})\}, \tag{9}$$

where $\mathbf{d}(\gamma^{(k)}) = (\partial\boldsymbol{\varphi}_{\mathcal{A}}(\gamma^{(k)})/\partial\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma^{(k)}))^{-1}\mathbf{v}_{\mathcal{A}}$. The predictor step of the PC algorithm developed to estimate the dgLASSO solution curve is based on Equation 7 with step size $\Delta\gamma = \min\{\Delta\gamma, \Delta\gamma^{\text{out}}\}$. A schematic overview of the PC algorithm is given in Table 1. Since the optimal step size is based on a local approximation, we also include an exclusion step for removing incorrectly included variables in the model. When an incorrect variable is included in the model after the corrector step, we have that there exists a non-active variable such that the absolute value of the corresponding Rao's score test statistic is greater than γ. Checking this is trivial. To overcome this drawback, the "optimal" step size from the previous step is reduced using a contractor factor cf.

Step	Algorithm
1	compute $\hat{\beta}_{a_0}$
2	$\mathcal{A} \leftarrow \arg\max_{a_j^c \in \mathcal{A}^c} \lvert r^u_{a_j^c}(\hat{\beta}_{a_0})\rvert$ and $\gamma \leftarrow \lvert r^u_{a_1}(\hat{\beta}_{a_0})\rvert$
3	repeat
4	use Equation 8 to compute $\Delta\gamma^{\text{in}}$ and set $\Delta\gamma \leftarrow \Delta\gamma^{\text{in}}$
5	if method = "dgLASSO" then
6	use Equation 9 to compute $\Delta\gamma^{\text{out}}$ and set $\Delta\gamma \leftarrow \min\{\Delta\gamma^{\text{in}}, \Delta\gamma^{\text{out}}\}$
7	set $\gamma \leftarrow \gamma - \Delta\gamma$
8	use Equation 7 to compute $\tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ (predictor step)
9	use $\tilde{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma)$ as starting point to solve System 6 (corrector step)
10	$\forall\, a_h^c \in \mathcal{A}^c$ compute $r^u_{a_h^c}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))$
11	if $\exists\, a_h^c \in \mathcal{A}^c$ such that $\lvert r^u_{a_h^c}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\gamma))\rvert > \gamma$ then
12	$\gamma \leftarrow \gamma \cdot cf$, with cf a small positive constant, and go to 8
13	update $\mathcal{A}$
14	until convergence criterion is met

Table 1: Pseudo-code of the predictor-corrector algorithm to compute the dgLARS/dgLASSO solution curve.

Cyclic coordinate descent (CCD) algorithm

In order to simplify our notation, in this subsection we drop the dependence of the estimate on the tuning parameter γ. Assume that $\hat{\boldsymbol{\beta}}$ is the point of the solution curve defined at $\gamma \in (\gamma^{(k+1)}; \gamma^{(k)}]$ by the dgLARS method. Considering System 6, it is easy to see that $\hat{\boldsymbol{\beta}}$ can be equivalently defined as solution of the following system of non-linear equations

$$\begin{cases} \partial_{a_0}\ell(\hat{\boldsymbol{\beta}}; \mathbf{y}) &= 0 \\ \partial_{a_1}\ell(\hat{\boldsymbol{\beta}}; \mathbf{y}) &= \gamma \cdot I^{1/2}_{a_1}(\hat{\boldsymbol{\beta}}) v_{a_1} \\ \quad\vdots & \quad\vdots \\ \partial_{a_k}\ell(\hat{\boldsymbol{\beta}}; \mathbf{y}) &= \gamma \cdot I^{1/2}_{a_k}(\hat{\boldsymbol{\beta}}) v_{a_k}, \end{cases}$$

where $v_{a_i} = \text{sign}(\hat{\beta}_{a_i})$. When we consider a predictor that is not included in the active set, say $a_h^c \in \mathcal{A}^c(\gamma)$, we have

$$\lvert \partial_{a_h^c}\ell(\hat{\boldsymbol{\beta}}; \mathbf{y})\rvert < \gamma \cdot I^{1/2}_{a_h^c}(\hat{\boldsymbol{\beta}}).$$

Suppose that we want to compute the value of the solution curve at $\gamma' = \gamma - \Delta\gamma$, where $\Delta\gamma > 0$. If $\Delta\gamma$ is small enough, we can approximate the gradient of the log-likelihood function as follows

$$\nabla\ell(\boldsymbol{\beta}; \mathbf{y}) \doteq \nabla\ell(\hat{\boldsymbol{\beta}}; \mathbf{y}) - I(\hat{\boldsymbol{\beta}})(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) = \mathbf{X}^\top \mathbf{W}(\hat{\boldsymbol{\beta}})(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}), \tag{10}$$

where $z_i = \hat{\eta}_i + \partial\eta(\hat{\mu}_i)/\partial\mu(y_i - \mu(\hat{\eta}_i))$ is the well-known working response variable and $\mathbf{W}(\hat{\boldsymbol{\beta}})$ is a diagonal matrix with ith element $w_i(\hat{\boldsymbol{\beta}}) = (\partial\mu(\hat{\eta}_i)/\partial\eta_i)^2 V^{-1}(\mu(\hat{\eta}_i))$. Approximation 10 is the basic step to relate the Fisher scoring algorithm with the iterative reweighted least squares (IRLS) algorithm. On the other hand, if $\Delta\gamma$ is small enough, the Fisher information matrix can be considered approximately constant. Then, as a consequence of the Karush-Kuhn-Tucker conditions, the value of the solution curve at γ' can be approximated by the

solution of the following constrained optimization problem,

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}) \left(z_i - \mathbf{x}_i^\top \boldsymbol{\beta} \right)^2 - \gamma' \sum_{m=1}^{p} I_m^{1/2}(\hat{\boldsymbol{\beta}}) |\beta_m|.$$

The previous expression suggests that, at the step t_1, $\hat{\boldsymbol{\beta}}(\gamma')$ can be updated by the following iterative rule

$$\hat{\boldsymbol{\beta}}^{(t_1+1)} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) \left(z_i^{(t_1)} - \mathbf{x}_i^\top \boldsymbol{\beta} \right)^2 - \gamma' \sum_{m=1}^{p} I_m^{1/2}(\hat{\boldsymbol{\beta}}^{(t_1)}) |\beta_m|. \tag{11}$$

Problem 11 can be efficiently solved using the same approach as proposed in Friedman, Hastie, and Tibshirani (2010). Suppose that we have estimated β_n, with $n \neq m$; in this case a coordinate descent step for solving Problem 11 is obtained by the solution of the following problem

$$\min_{\beta_m} \frac{1}{2} \sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) \left(r_{i,m}^{(t_1)} - x_{im}\beta_m \right)^2 - \gamma' I_m^{1/2}(\hat{\boldsymbol{\beta}}^{(t_1)}) |\beta_m| - \gamma' \sum_{n \neq m}^{p} I_n^{1/2}(\hat{\boldsymbol{\beta}}^{(t_1)}) |\beta_n|,$$

where $r_{i,m}^{(t_1)} = z_i^{(t_1)} - \sum_{n \neq m}^{p} x_{in} \hat{\beta}_n^{(t_1)}$ is the partial residual for fitting β_m. It is easy to see that, at the step t_2, the coordinate-wise updating rule has the following form

$$\hat{\beta}_m^{(t_1,t_2+1)} = \frac{S\left(\sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) r_{i,m}^{(t_1)} x_{im}; \gamma' \cdot I_m^{1/2}(\hat{\boldsymbol{\beta}}^{(t_1)}) \right)}{\sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) x_{im}^2},$$

where $S(x;\lambda) = \text{sign}(x)(|x| - \lambda)_+$ is the soft-thresholding operator. The pseudo-code of the proposed CCD algorithm is reported in Table 2. In order to reduce the computational burden, in our algorithm we initially cycle only through the predictors that are included in the current active set $\mathcal{A}$ and then, when the convergence is met, we check if the active set is changed.

2.3. Generalized degrees of freedom

The behavior of the dgLARS method is closely related to the way of selecting the optimal value of the tuning parameter γ. For the LASSO estimator, Zou, Hastie, and Tibshirani (2007) developed an adaptive model selection criterion to select the regularization parameter based on a rigorous definition of the degrees of freedom.

In order to define an adaptive model selection criterion, Augugliaro *et al.* (2013) propose the notion of generalized degrees of freedom (GDF) for the dgLARS method. This notion comes from a differential geometric approach to the covariance penalty theory (Efron 2004), which defines the degrees of freedom for a general modeling procedure. When it is possible to compute the MLE of the considered GLM, the authors also propose a possible estimator of the GDF that is implemented in the package by the function `gdf()` discussed in Section 3.2. When we work with a logistic regression model, it has been shown by a simulation study that the proposed estimator is better than the number of non-zero estimated coefficients. On the other hand, simulation studies show that for a Poisson regression model the number of non-zero estimated coefficients can be considered a good estimator of the GDF. See Augugliaro

Step	Algorithm
1	$\hat{\boldsymbol{\beta}}^{(t_1)} \leftarrow \hat{\boldsymbol{\beta}}(\gamma)$
2	$t_1 \leftarrow 1$ and $flag \leftarrow 0$
3	repeat
4	compute $\mathbf{W}(\hat{\boldsymbol{\beta}}^{(t_1)}), \mathbf{z}^{(t_1)}$ and $I_m(\hat{\boldsymbol{\beta}}^{(t_1)})$
5	$t_2 \leftarrow 0$
6	repeat
7	$t_2 \leftarrow t_2 + 1$
8	do $m = 1$ to $\lvert\mathcal{A}\rvert$
9	$a_i \leftarrow \mathcal{A}(m)$
10	compute $r_{i,a_i}^{(t_1)}$ and $\sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) r_{i,a_i}^{(t_1)} x_{ia_i}$
11	$\hat{\beta}_{a_i}^{(t_1,t_2)} \leftarrow S\left(\sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) r_{i,a_i}^{(t_1)} x_{ia_i}; \gamma' \cdot I_{a_i}^{1/2}(\hat{\boldsymbol{\beta}}^{(t_1)})\right) \Big/ \sum_{i=1}^{N} w_i(\hat{\boldsymbol{\beta}}^{(t_1)}) x_{ia_i}^2$
12	end do
13	until convergence is met
14	$t_1 \leftarrow t_1 + 1$ and $\hat{\boldsymbol{\beta}}^{(t_1)} \leftarrow \hat{\boldsymbol{\beta}}^{(t_1,t_2)}$
15	until convergence is met
16	do $m = 1$ to $\lvert\mathcal{A}^c\rvert$
17	$a_h^c \leftarrow \mathcal{A}^c(m)$
18	compute $r_{a_h^c}^u(\hat{\boldsymbol{\beta}}^{(t_1)})$
19	if $\lvert r_{a_h^c}^u(\hat{\boldsymbol{\beta}}^{(t_1)})\rvert \geq \gamma'$ then
20	$\mathcal{A} \leftarrow a_h^c$ and $flag \leftarrow 1$
21	end if
22	end do
23	if $flag = 1$ then go to step 2
24	else $\hat{\boldsymbol{\beta}}(\gamma') \leftarrow \hat{\boldsymbol{\beta}}^{(t_1)}$
25	end if

Table 2: Pseudo-code of the CCD method proposed to compute the solution curve implicitly defined by the dgLARS method.

et al. (2013) for more details. However, the derivation of an estimator of the generalized degrees of freedom in a high-dimensional setting GLM is still an open question and subject of further research.

3. The dglars package

The **dglars** package is an R package containing a collection of tools related to the dgLARS method. In the following of this section we describe the functions available with this package.

3.1. Description of the `dglars()` and `dglars.fit()` functions

The main functions of this package are `dglars()` and `dglars.fit()`. The first one

```
dglars(formula, family = c("binomial", "poisson"), data, subset,
  contrast = NULL, control = list())
```

is a wrapper function implemented to handle the formula interface usually used in R to create the $N \times p$-dimensional design matrix `X` and the N-dimensional response vector `y`. These objects, together with the arguments `family` and `control`, are passed to the function `dglars.fit()`

```
dglars.fit(X, y, family = c("binomial", "poisson"), control = list())
```

which is the R function used to compute the dgLARS/dgLASSO solution curve. Although in R the formula interface is the more familiar way to specify the linear predictor in a GLM, in a high dimensional setting this management of the involved model variables can be inefficient from a computational point of view. For this reason we recommend using the function `dglars.fit()` directly for simulation studies and real applications in the cases where p is very large.

The argument `control` is a named list of control parameters passed to the two algorithms described in Section 2.2. In order to reduce the computational time needed to compute the dgLARS/dgLASSO solution curve, the two algorithms are written in Fortran 90 and handled by two specific R wrapper functions called `dglars_pc()` and `dglars_ccd()`. These two functions are not user-intended and for this reason they are not described in this paper. By default, `control` argument is defined as follows

```
control = list(algorithm = "pc", method = "dgLASSO", np = NULL,
  g0 = NULL, eps = 1.0e-05, nv = NULL, dg_max = 0, nNR = 50,
  NReps = 1.0e-06, ncrct = 50, cf = 0.5, nccd = 1.0e+05)
```

Using the control parameter `algorithm` it is possible to select the algorithm used to fit the dgLARS solution curve, i.e., setting `algorithm = "pc"` the default PC algorithm is used, whereas the CCD algorithm is used when `algorithm = "ccd"` is selected. The group of control parameters `method`, `np`, `g0` and `eps`, is composed of those elements that are shared by the two algorithms. The argument `method` is used to choose between the dgLASSO solution curve (`method = "dgLASSO"`) and the dgLARS solution curve (`method = "dgLARS"`), while `np` is used to define the maximum number of points on the solution curve. As seen in Section 2.2, since the PC algorithm can compute the step size by a local approximation, the number of effective points of the solution curve can be significantly smaller than `np`. In contrast, the CCD algorithm fits the dgLARS solution curve using a multiplicative grid of `np` values of the tuning parameter. The `g0` control parameter is used to define the smallest value of the tuning parameter. By default this parameter is set to `1.0e-04` when $p > N$ and to 0.05 otherwise. Finally, `eps` is used to assess the convergence of the two algorithms. When the PC algorithm is used, `eps` is also used to identify a predictor that will be included in the active set, namely when the absolute value of the corresponding Rao's score test statistic belongs to $[\gamma - \texttt{eps}; \gamma + \texttt{eps}]$.

The group composed by `nv`, `dg_max`, `nNR`, `NReps`, `ncrct` and `cf` contains the control parameters specific to the PC algorithm. `nv` is used to define the maximum number of predictors included in the model, while `dg_max` is used to fix the step size. When setting `dg_max = 0` (default) the PC algorithm uses the local approximation to compute the γ value to evaluate the inclusion or

exclusion of a predictor from the active set. The control parameters `nNR` and `NReps` are used to set the number of steps and to define the convergence of the Newton-Raphson algorithm used in the corrector step. When the Newton-Raphson algorithm does not converge or when there exists a predictor such that the absolute value of the corresponding Rao's score test statistic is greater than γ+`eps`, the step size is reduced by the contractor factor `cf`, i.e., $\Delta\gamma = \Delta\gamma \cdot$ `cf`, and then the corrector step is repeated. The control parameter `ncrct` sets the maximum number of attempts for the corrector step.

Finally, the control parameter `nccd` is used to define the maximum number of steps of the CCD algorithm.

An example of a simulated logistic regression model

To gain more insight on how to use the `dglars()` function we consider a simulated data set. First we load the **dglars** package in the R session by the code

```
R> library("dglars")
```

then we simulate a data set from a logistic regression model with sample size equal to 100 and $p = 4$ predictors. We also assume that only the first two predictors influence the response variable. The used R code is given by:

```
R> set.seed(321)
R> n <- 100
R> p <- 4
R> s <- 2
R> X <- matrix(rnorm(n * p), n, p)
R> bs <- rep(1, s)
R> Xs <- X[, 1:s]
R> eta <- drop(1 + drop(Xs %*% bs))
R> mu <- binomial()$linkinv(eta)
R> y <- rbinom(n, 1, mu)
R> dataset <- data.frame(y = y, X = X)
R> out_dglasso_pc <- dglars(y ~ ., family = "binomial", data = dataset)
```

The last command of the previous code stands, as usual, for

```
R> out_dglasso_pc <- dglars(y ~ X.1 + X.2 + X.3 + X.4, family = "binomial",
+    data = dataset)
```

`dglars()` and `dglars.fit()` return an object of S3 class '`dglars`'.

```
R> class(out_dglasso_pc)
```

```
[1] "dglars"
```

which is a list containing a matrix named `beta` used to store the estimated points of the solution curve, the vector `dev` of deviances, the vector `g` containing the sequence of the used

values of the tuning parameter and the vector `df` containing the number of non-zero estimated coefficients including the intercept.

By default, `dglars()` computes the dgLASSO solution curve; the dgLARS solution curve can be computed using the control parameter `method`, i.e.,

```
R> out_dglars_pc <- dglars(y  ~ ., family = "binomial", data = dataset,
+    control = list(method = "dgLARS"))
```

The `print()` method for the '`dglars`' object can be used to print the basic information, i.e., the call that produced the '`dglars`' object with a five-column table showing the names of predictors included or excluded from the active set, the sequence of γ values used to compute the dgLARS solution curve and the corresponding deviance and fraction of explained deviance, respectively. The number of non-zero estimated coefficients is also reported.

By printing the '`dglars`' object `out_dglasso_pc` for our simulated data set, we can see that the dgLASSO method first finds the true predictors and then includes the other false predictors.

```
R> out_dglasso_pc

Call:  dglars(formula = y ~ ., family = "binomial", data = dataset)

  Sequence       g     Dev     %Dev  df
           3.6372  122.17  0.00000   1
     +X.2
           3.4263  120.70  0.01203   2
           3.3221  120.00  0.01779   2
           3.2703  119.66  0.02060   2
           3.2445  119.49  0.02199   2
           3.2316  119.40  0.02268   2
           3.2187  119.32  0.02337   2
     +X.1
           2.0552  107.17  0.12278   3
           1.4839  102.78  0.15872   3
           1.2050  101.08  0.17262   3
           1.0677  100.36  0.17851   3
           0.9996  100.04  0.18120   3
           0.9657   99.88  0.18247   3
           0.9488   99.80  0.18309   3
           0.9319   99.73  0.18370   3
     +X.4
           0.8109   98.93  0.19022   4
           0.8109   98.93  0.19022   4
     +X.3
           0.0001   95.70  0.21667   5

Algorithm pc ( method = dgLASSO ) with exit = 0
```

To be more specific, at $\gamma^{(1)} = 3.6372$, i.e., the starting value of the Rao's score test statistic, the predictor `X.2` has the smallest angle with the tangent residual vector and is then included in the active set. The predictor `X.1` is included in the active set at $\gamma^{(2)} = 3.2187$, this means that for any $\gamma \in [\gamma^{(2)}; \gamma^{(1)})$ only the intercept term and the regression coefficient $\hat{\beta}_2(\gamma)$ are different from zero and the number of non-zero estimates is equal to 2. The third predictor is included at $\gamma^{(3)} = 0.9319$, which means that for any $\gamma \in [\gamma^{(3)}; \gamma^{(2)})$ the number of non-zero estimates is equal to 3, i.e the intercept term and the regression coefficients $\hat{\beta}_1(\gamma)$ and $\hat{\beta}_2(\gamma)$. This process continues until γ is equal to the control argument `g0`. The estimated coefficient path can be extracted using the `coef()` method available for the '`dglars`' object. For example, with the following R code we can see the sequence of the first ten values of the tuning parameter γ with the corresponding estimated points.

```
R> g <- out_dglasso_pc$g
R> coef_path <- coef(out_dglasso_pc)
R> path <- rbind(g, coef_path)[, 1:10]
R> print(path, digits = 3)
```

```
      [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]  [,8]  [,9] [,10]
g    3.637 3.4263 3.3221 3.2703 3.2445 3.2316 3.2187 2.055 1.484 1.205
Int. 0.847 0.8478 0.8483 0.8487 0.8489 0.8490 0.8491 0.885 0.927 0.954
X.1  0.000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.261 0.405 0.480
X.2  0.000 0.0498 0.0748 0.0873 0.0935 0.0967 0.0998 0.387 0.547 0.630
X.3  0.000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.000 0.000 0.000
X.4  0.000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.000 0.000 0.000
```

More information about the estimated sequence of models can be obtained using the `summary()` method for the '`dglars`' object, i.e.,

```
summary.dglars(object, k = c("BIC", "AIC"), complexity = c("df", "gdf"),
  digits = max(3, getOption("digits") - 3), ...)
```

where `object` is a fitted '`dglars`' object. To choose the best solution point, the `summary()` method computes the measure of goodness of fit

$$\text{residual deviance} + k \times \text{complexity}, \tag{12}$$

where k is a non-negative value used to weight the complexity of the fitted model. Using the argument `complexity` the user can choose between two different definitions of complexity of a fitted model, i.e., the well known number of estimated non-zero coefficients (`complexity = "df"`) and the notion of generalized degrees of freedom (`complexity = "gdf"`). More details about the notion of generalized degrees of freedom are given in Sections 2.3 and 3.2. Setting `k = "BIC"` and `complexity = "df"`, which are the default values, Definition 12 shows that the `summary` method for '`dglars`' objects reports the Bayesian information criterion (BIC; Schwarz 1978). The Akaike information criterion (AIC; Akaike 1973) can be easily computed setting `k = "AIC"` and `complexity = "df"`. The user can also define own measures of goodness of fit setting `k` as any non-negative value.

The following R code shows that the output printed by the `summary()` method is divided in two different sections.

```
R> summary(out_dglasso_pc, k = "BIC", complexity = "df")

Call:  dglars(formula = y ~ ., family = "binomial", data = dataset)

  Sequence       g     Dev  df    BIC   Rank
           3.6372  122.17   1  126.8  12
      +X.2
           3.4263  120.70   2  129.9  18
           3.3221  120.00   2  129.2  17
           3.2703  119.66   2  128.9  16
           3.2445  119.49   2  128.7  15
           3.2316  119.40   2  128.6  14
           3.2187  119.32   2  128.5  13
      +X.1
           2.0552  107.17   3  121.0  11
           1.4839  102.78   3  116.6   7
           1.2050  101.08   3  114.9   6
           1.0677  100.36   3  114.2   5
           0.9996  100.04   3  113.9   4
           0.9657   99.88   3  113.7   3
           0.9488   99.80   3  113.6   2
           0.9319   99.73   3  113.5   1 <-
      +X.4
           0.8109   98.93   4  117.4   9
           0.8109   98.93   4  117.4   8
      +X.3
           0.0001   95.70   5  118.7  10

============================================

Best model identified by BIC criterion ( k = 4.60517 and complexity = df ):

 y ~ X.1 + X.2

Coefficients:

  Int.     X.1     X.2
0.9854  0.5571  0.7157


 BIC :  113.5

===

Algorithm pc ( method = dgLASSO ) with exit = 0
```

The first section completes the basic information printed out by the default `print()` method

showing the BIC. The ranking of the estimated models obtained by this measure of goodness of fit is also shown and the corresponding best model is identified by an arrow on the right. The second section shows the formula of the identified best model and the corresponding estimated coefficients. From the previous output we can see that the best model identified by the BIC criterion is that one with only the predictors `X.1` and `X.2`, which are the two true predictors that influence the response variable in our simulated model.

The following R code shows the results given by the `summary()` method when the AIC is used as measure of goodness of fit.

```
R> summary(out_dglasso_pc, k = "AIC", complexity = "df")

Call:  dglars(formula = y ~ ., family = "binomial", data = dataset)

  Sequence       g     Dev  df    AIC   Rank
           3.6372  122.17   1  124.2  17
      +X.2
           3.4263  120.70   2  124.7  18
           3.3221  120.00   2  124.0  16
           3.2703  119.66   2  123.7  15
           3.2445  119.49   2  123.5  14
           3.2316  119.40   2  123.4  13
           3.2187  119.32   2  123.3  12
      +X.1
           2.0552  107.17   3  113.2  11
           1.4839  102.78   3  108.8  10
           1.2050  101.08   3  107.1   9
           1.0677  100.36   3  106.4   6
           0.9996  100.04   3  106.0   5
           0.9657   99.88   3  105.9   4
           0.9488   99.80   3  105.8   3
           0.9319   99.73   3  105.7   2
      +X.4
           0.8109   98.93   4  106.9   8
           0.8109   98.93   4  106.9   7
      +X.3
           0.0001   95.70   5  105.7   1 <-

================================================

Best model identified by AIC criterion ( k = 2 and complexity = df ):

 y ~ X.1 + X.2 + X.3 + X.4

Coefficients:

   Int.      X.1      X.2      X.3      X.4
 1.1959   0.8573   1.1008  -0.1764  -0.2847
```

```
 AIC :  105.7

===

Algorithm pc ( method = dgLASSO ) with exit = 0
```

In this case we can see that the best model contains all the considered predictors, even if the second best model is the same model identified by using the BIC criterion.

The user can plot the output from the `dglars()` function using the `plot()` method for the '`dglars`' object, i.e.,

```
plot(x, k = c("BIC", "AIC"), complexity = c("df", "gdf"),
  g.gof = NULL, ...)
```

where `x` is a fitted '`dglars`' object while the arguments `k` and `complexity` are equal to the arguments of the `summary` method for '`dglars`' objects. With the following R code

```
R> out_dglasso_pc2 <- dglars(y ~ ., family = "binomial", data = dataset,
+    control = list(dg_max = 0.1))
R> par(mfrow = c(2, 3))
R> plot(out_dglasso_pc2, k = "BIC", complexity = "df")
R> plot(out_dglasso_pc2, k = "AIC", complexity = "df")
```

we first reduce the step size setting the control parameter `dg_max = 0.1` and then we plot the output from the `dglars()` function. As we have done for the `summary()` method, we first use the BIC criterion to select the best model and then we use the AIC criterion. As shown in Figure 2, when we fit the dgLARS solution curve using the PC algorithm, the `plot()` method produces three different plots, namely the plots showing the sequence of the BIC (first row) or AIC (second row) as function of γ and the plots showing the paths of the coefficients and of the Rao's score test statistics. The last plot is not available when the dgLARS solution curve is fitted using the CCD algorithm. The values of the tuning parameter corresponding to a change in the active set are identified by vertical dashed gray lines, while the optimal value of the tuning parameter γ, according to the BIC or AIC, is identified by a red dashed line. Using the argument `g.gof` the user can specify a value of the tuning parameter to identify different best models. Finally the argument `...` can be used to specify additional graphical parameters.

Comparison between PC and CCD algorithm

As we have seen in the previous section, the **dglars** package implements two different algorithms to compute the dgLARS solution curve., i.e., a PC method and a CCD algorithm. Although the two algorithms compute the same solution curve the results can look different. To gain more insight we consider the following R code

```
R> set.seed(321)
R> n <- 100
```

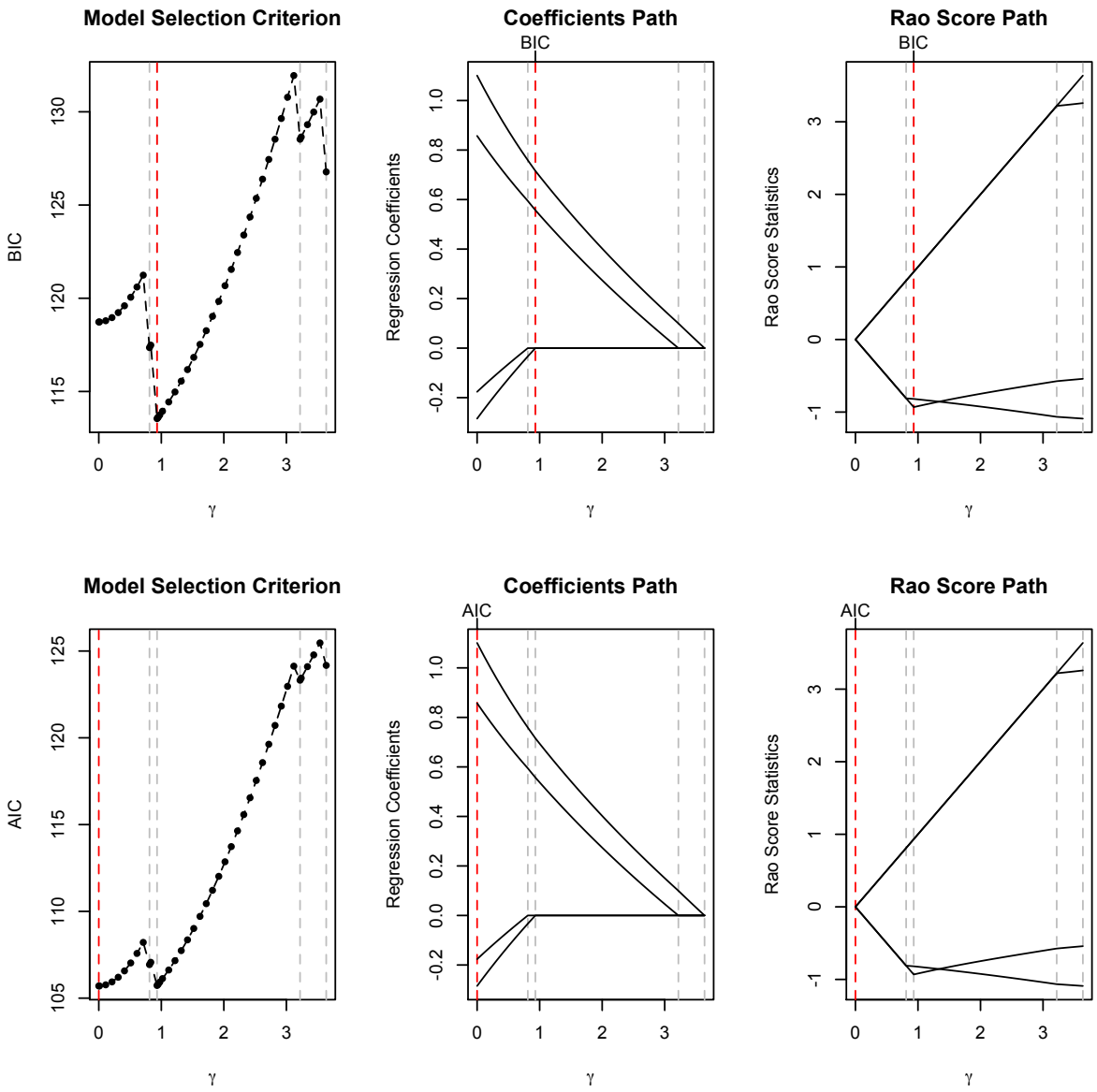

Figure 2: Plot of the path of the coefficients, of the Rao's score test statistics and the computed values of BIC and AIC for the simulated logistic regression model.

```
R> p <- 5
R> X <- matrix(rnorm(n * p), n, p)
R> b <- 1:2
R> eta <- b[1] + X[, 1] * b[2]
R> mu <- binomial()$linkinv(eta)
R> y <- rbinom(n, 1, mu)
R> dataset <- data.frame(y = y, X = X)
```

where we simulate a logistic regression model with sample size equal to 100 and 5 predictors; only the first two predictors affect the response variable. By the following code

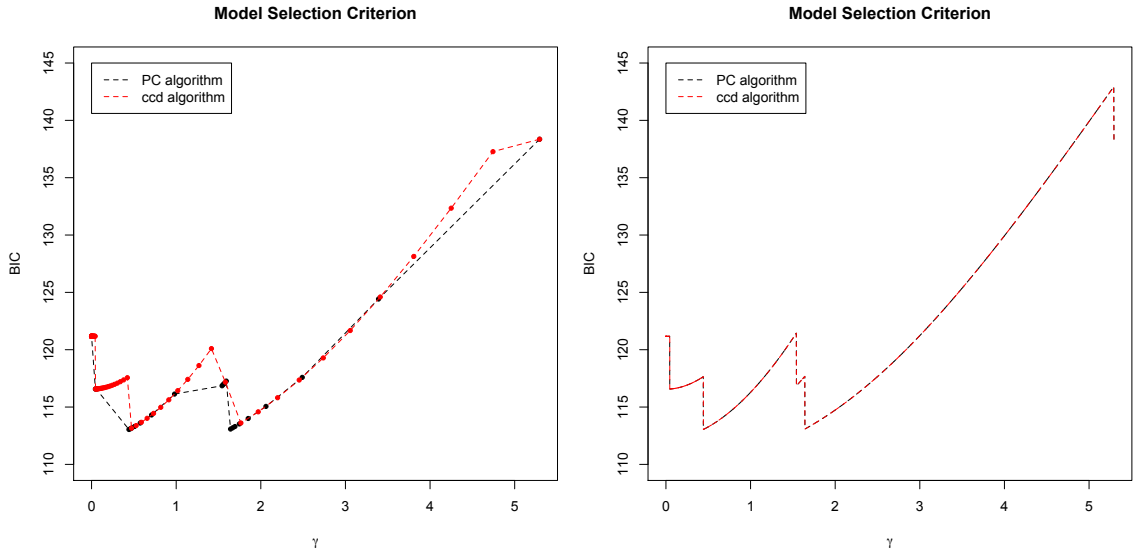

Figure 3: Paths of the BIC values computed using the PC algorithm (black dashed lines) and the CCD method (red dashed lines). The left panel is obtained using the default values of the two algorithms, while the right panel is obtained using a finer grid for the γ values.

```
R> out_dglasso_pc <- dglars(y ~ ., family = "binomial", data = dataset)
R> out_dglasso_ccd <- dglars(y ~ ., family = "binomial", data = dataset,
+    control = list(algorithm = "ccd"))
```

we estimate the dgLASSO solution curve using the PC and the CCD algorithm, respectively. The left panel of Figure 3, obtained by the following code, shows the path of the BIC values computed using the PC algorithm (black dashed line) and the CCD algorithm (red dashed line).

```
R> bic_pc <- out_dglasso_pc$dev + log(n) * out_dglasso_pc$df
R> g_pc <- out_dglasso_pc$g
R> bic_ccd <- out_dglasso_ccd$dev + log(n) * out_dglasso_ccd$df
R> g_ccd <- out_dglasso_ccd$g
R> plot(g_pc, bic_pc, type = "o", pch = 20, lty = 2, ylim = c(110, 145),
+    xlab = expression(gamma), ylab = "BIC",
+    main = "Model Selection Criterion")
R> points(g_ccd, bic_ccd, type = "o", pch = 20, lty = 2, col = 2)
R> legend(x = 0, y = 145, legend = c("PC algorithm", "ccd algorithm"),
+    col = c(1, 2), lty = 2)
```

We can clearly see that the two paths seem to be quite different. This difference is simply due to the method used to compute the γ values, i.e., the PC algorithm approximates the γ value corresponding to a change in the active set, while the CCD algorithm uses an equispaced sequence of $\log(\gamma)$ values. This means that we can remove the difference observed in the left panel of Figure 3 simply by using a finer grid of γ values. As shown by the following code, this can be done using the control parameters `dg_max` and `np`, i.e.,

```
R> out_dglasso_pc <- dglars(y ~ ., family = "binomial", data = dataset,
+    control = list(dg_max = 1.0e-03, np = 1.0e+04))
R> out_dglasso_ccd <- dglars(y ~ ., family = "binomial", data = dataset,
+    control = list(algorithm = "ccd", np = 1.0e+04))
```

The right panel of Figure 3, obtained by the code

```
R> bic_pc <- out_dglasso_pc$dev + log(n) * out_dglasso_pc$df
R> g_pc <- out_dglasso_pc$g
R> bic_ccd <- out_dglasso_ccd$dev + log(n) * out_dglasso_ccd$df
R> g_ccd <- out_dglasso_ccd$g
R> plot(g_pc, bic_pc, type = "l", pch = 20, lty = 2, ylim = c(110, 145),
+    xlab = expression(gamma), ylab = "BIC",
+    main = "Model Selection Criterion")
R> points(g_ccd, bic_ccd, type = "l", pch = 1, lty = 2, col = 2)
R> legend(x = 0, y = 145, legend = c("PC algorithm", "ccd algorithm"),
+    col = c(1, 2), lty = 2)
```

shows that the difference previously observed between the two paths is now removed. The same difference can be observed in coefficient path estimated by the two algorithms, but also in this case this difference can be removed using a finer grid for the γ values. From a computational point of view, the main consequence of using a finer grid is an increase in the run times needed to compute the two solution curves.

As we have previously said, the main problem of the PC algorithm is related to the number of arithmetic operations needed to compute the Euler predictor, which requires the inversion of an adequate Jacobian matrix. The effects stemming from this problem can be shown by the following R code

```
R> set.seed(321)
R> n <- 100
R> p <- c(10, 100)
R> s <- 2
R> X <- matrix(rnorm(n * p[2]), n, p[2])
R> bs <- rep(2, s)
R> Xs <- X[, 1:s]
R> eta <- drop(1 + drop(Xs %*% bs))
R> mu <- binomial()$linkinv(eta)
R> y <- rbinom(n, 1, mu)
R> results.time <- array(0, dim = c(2, 3, 2),
+    dimnames = list(algorithm = c("pc", "ccd"),
+    time = c("user", "system", "elapsed"), p = p))
R> results.time["pc", , "10"] <- system.time(dglars.fit(X = X[, 1:p[1]],
+    y = y, family = "binomial",
+    control = list(algorithm = "pc", eps = 1e-3, g0 = 0.1)))[1:3]
R> results.time["ccd", , "10"] <- system.time(dglars(X = X[, 1:p[1]],
+    y = y, family = "binomial",
+    control = list(algorithm = "ccd", eps = 1e-3, g0 = 0.1)))[1:3]
```

```
R> results.time["pc", , "100"] <- system.time(dglars.fit(X = X[, 1:p[2]],
+    y = y, family = "binomial",
+    control = list(algorithm = "pc", eps = 1e-3, g0 = 0.1)))[1:3]
R> results.time["ccd", , "100"] <- system.time(dglars.fit(X = X[, 1:p[2]],
+    y = y, family = "binomial",
+    control = list(algorithm = "ccd", eps = 1e-3, g0 = 0.1)))[1:3]
R> results.time

, , p = 10

          time
algorithm  user system elapsed
      pc  0.012  0.001   0.013
      ccd 0.010  0.000   0.010

, , p = 100

          time
algorithm  user system elapsed
      pc  0.267  0.039   0.305
      ccd 0.052  0.002   0.053
```

where we use the control parameter `algorithm` of the `dglars.fit()` function to choose between the algorithms described in Tables 1 and 2. The values of the control parameters `g0` and `eps` are chosen in order to remove the instability coming from the link function in a high-dimensional setting.

In order to better understand the effects of the number of predictors on the run times of the two algorithms, we use a simple simulation study based on a logistic regression model with sample size equal to $N = (100, 200, 300)$ and $p = (25, 50, 100, 200, 400, 600, 800, 1000)$. The design matrix is obtained as follows

(a) $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_p$ sampled from a $N(\mathbf{0}; \Sigma)$, where the diagonal and off-diagonal elements of Σ are 1 and 0, respectively;

(b) the same of (a) but with $\mathsf{COR}(\mathcal{X}_i; \mathcal{X}_j) = \rho^{|i-j|}$ with $\rho = 0.9$.

To simulate the binary response vector we use a model with intercept and with only three predictors; the corresponding non-zero coefficients were chosen equal to 2. In Table 3 we report the average CPU times in seconds coming from 100 simulation runs. All timings reported were carried out on a personal computer with AMD Athlon II X2 250 dual-core processor. We clearly see that the proposed CCD algorithm is always significantly faster than the PC algorithm. The difference between the two algorithms is greater when we consider the scenario with high correlation among the predictors. In this case the PC algorithm dramatically increases its computational time, while the CCD algorithm slows down just a little in this scenario. In Figure 4 we show the average CPU times of the considered algorithms for $N = 300$ for both scenarios (a) and (b).

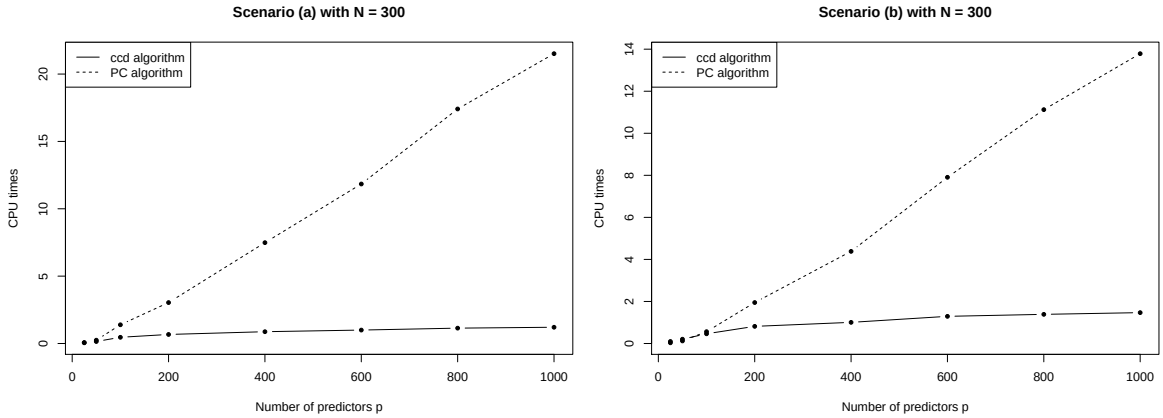

Figure 4: CPU times in seconds for the CCD algorithm and the PC algorithm recorded in scenario (a) and (b) with sample size $N = 300$.

3.2. Description of the `gdf()` function

As we have discussed in Section 2.3, the behavior of the dgLARS method, as a variable selection method, is closely related to the way we define the complexity of a model. When we have a Poisson regression model, the number of non-zero estimated coefficients can be considered a good approximation of the GDF, but when we have a logistic regression model, Augugliaro *et al.* (2013) propose another possible estimator of the GDF. In the same paper the authors show how the proposed GDF has a better behavior than using the number of non-zero estimated coefficients.

The proposed estimator is implemented by the function `gdf()` with argument

```
gdf(object)
```

where `object` is a fitted '`dglars`' object. `gdf()` is called by the `summary` method for '`dglars`' objects

```
summary(object, k = c("BIC", "AIC"), complexity = c("df", "gdf"),
  digits = max(3, getOption("digits") - 3), ...)
```

when the argument `complexity` is set to `"gdf"`.

As shown in Augugliaro *et al.* (2013), the proposed estimator of the GDF requires the dgLARS estimates and the MLE of the GLM specified using all the predictors which, in some cases, cannot be computed. For this reason we prefer to use the number of non-zero estimated coefficients as default value for the argument `complexity`.

An example of use for a simulated logistic regression model: Continued

To gain more insight about the main differences between the proposed GDF estimator and the number of non-zero estimated coefficients, we consider the simulated logistic regression model used in Section 3.1. The following R code is used to show the first ten estimates given by the proposed estimator (`gdf`) and by the number of non-zero estimated coefficients (`df`).

		N					
		100		200		300	
Scenario	p	CCD	PC	CCD	PC	CCD	PC
(a)	1000	0.254	2.091	0.618	8.284	1.206	21.513
		(0.051)	(0.510)	(0.083)	(1.504)	(0.128)	(3.794)
	800	0.249	1.922	0.584	7.148	1.140	17.409
		(0.040)	(0.380)	(0.067)	(1.240)	(0.118)	(3.229)
	600	0.200	1.238	0.612	5.917	0.997	11.840
		(0.034)	(0.232)	(0.079)	(1.003)	(0.122)	(2.319)
	400	0.153	0.712	0.459	3.265	0.875	7.485
		(0.030)	(0.146)	(0.057)	(0.570)	(0.089)	(1.433)
	200	0.144	0.389	0.387	1.524	0.671	3.038
		(0.025)	(0.087)	(0.046)	(0.326)	(0.067)	(0.634)
	100	0.102	0.192	0.272	0.602	0.461	1.390
		(0.019)	(0.042)	(0.031)	(0.134)	(0.071)	(0.231)
	50	0.078	0.071	0.133	0.189	0.148	0.246
		(0.009)	(0.018)	(0.037)	(0.034)	(0.038)	(0.043)
	25	0.037	0.026	0.053	0.040	0.071	0.052
		(0.010)	(0.005)	(0.013)	(0.008)	(0.013)	(0.009)
(b)	1000	0.284	1.394	0.833	6.801	1.469	13.786
		(0.081)	(0.357)	(0.173)	(1.562)	(0.295)	(3.390)
	800	0.274	1.224	0.736	4.841	1.387	11.124
		(0.075)	(0.302)	(0.146)	(1.319)	(0.300)	(2.638)
	600	0.225	0.637	0.707	3.785	1.292	7.906
		(0.072)	(0.182)	(0.155)	(0.813)	(0.310)	(1.853)
	400	0.191	0.485	0.580	2.213	1.006	4.384
		(0.056)	(0.137)	(0.149)	(0.455)	(0.204)	(0.977)
	200	0.178	0.281	0.344	0.693	0.816	1.949
		(0.059)	(0.069)	(0.094)	(0.162)	(0.148)	(0.412)
	100	0.123	0.142	0.291	0.354	0.463	0.563
		(0.040)	(0.025)	(0.054)	(0.063)	(0.105)	(0.109)
	50	0.083	0.037	0.151	0.086	0.205	0.123
		(0.024)	(0.009)	(0.046)	(0.020)	(0.055)	(0.026)
	25	0.046	0.014	0.071	0.025	0.097	0.034
		(0.012)	(0.003)	(0.015)	(0.006)	(0.022)	(0.008)

Table 3: Average CPU times in seconds to compute the solution curve using the CCD algorithm and the PC algorithm. Standard deviations are reported in parentheses.

```
R> out_dglasso_pc <- dglars(y ~ ., family = "binomial", data = dataset,
+    control = list(g0 = 0))
R> df <- out_dglasso_pc$df
R> bh <- coef(out_dglasso_pc)
R> gdfh <- gdf(out_dglasso_pc)
R> complexity <- rbind(df, gdfh)
R> rownames(complexity) <- c("df", "gdf")
```

```
R> print(complexity[, 1:10], digits = 3)

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
df  1.000 2.00 2.00 2.00 2.00 2.00 2.00 3.00 3.00  3.00
gdf 0.749 1.35 1.34 1.34 1.34 1.34 1.34 2.09 2.20  2.28
```

We can see that the main difference is that the estimates of the `gdf` are always smaller than `df`. This behavior finds a theoretical justification by the results given in Efron (1986), in other words, the number of non-zero estimated coefficients is equal to the `gdf` only when we are evaluating the complexity of a model on a point that is estimated by using the maximum likelihood method. When we set the argument `complexity = "gdf"`, the `summary()` method uses the proposed estimator of the GDF to define the BIC or AIC values.

Coming back to our simulated logistic regression model, the following R code

```
R> summary(out_dglasso_pc, k = "AIC", complexity = "gdf")

Call:  dglars(formula = y ~ ., family = "binomial", data = dataset,
    control = list(g0 = 0))

  Sequence           g     Dev     gdf    AIC   Rank
           3.637e+00  122.17  0.7487  123.7  18
      +X.2
           3.426e+00  120.70  1.3480  123.4  17
           3.322e+00  120.00  1.3449  122.7  16
           3.270e+00  119.66  1.3436  122.3  15
           3.245e+00  119.49  1.3430  122.2  14
           3.232e+00  119.40  1.3427  122.1  13
           3.219e+00  119.32  1.3424  122.0  12
      +X.1
           2.055e+00  107.17  2.0856  111.3  11
           1.484e+00  102.78  2.1957  107.2  10
           1.205e+00  101.08  2.2758  105.6   8
           1.068e+00  100.36  2.3224  105.0   5
           9.996e-01  100.04  2.3473  104.7   4
           9.657e-01   99.88  2.3602  104.6   3
           9.488e-01   99.80  2.3667  104.5   2
           9.319e-01   99.73  2.3734  104.5   1 <-
      +X.4
           8.109e-01   98.93  3.2726  105.5   7
           8.109e-01   98.93  3.2726  105.5   6
      +X.3
           3.212e-13   95.70  5.0000  105.7   9

================================================

Best model identified by AIC criterion ( k = 2 and complexity = gdf ):
```

```
 y ~ X.1 + X.2

Coefficients:

  Int.     X.1     X.2
0.9854  0.5571  0.7157


 AIC :  104.5

===

Algorithm pc ( method = dgLASSO ) with exit = 0
```

shows that the AIC, defined using the estimates of the GDF, selects the logistic regression model with the true predictors. This result is in contrast to what we have previously obtained, since in the first application of the `summary()` method, the AIC identified a logistic regression model with four predictors. The true logistic regression model is also identified if we use the BIC based on the GDF. This result is not reported for sake of brevity. However, we point out that BIC defined using the estimator of GDF tends usually to select sparser models than AIC defined also with the estimator of GDF.

3.3. Description of the `cvdglars()` and `cvdglars.fit()` functions

In the previous section we have seen that we can use the `summary` method for '`dglars`' objects to select the optimal value of the tuning parameter γ by using an information criterion, such as the BIC or AIC. When the user has enough data, another possible way to estimate the solution point of the dgLARS solution curve is by k-fold cross-validation (Hastie, Tibshirani, and Friedman 2009).

In the **dglars** package, the k-fold cross-validation is implemented by the functions `cvdglars()` and `cvdglars.fit()`. The first one can be used with arguments

```
cvdglars(formula, family = c("binomial", "poisson"), data,
  contrast = NULL, subset, control = list())
```

As we have done for `dglars()`, this function is a wrapper function of the more efficient function `cvdglars.fit()`. It is implemented only to handle the classical R formula interface but it can be computational inefficient in a high-dimensional setting. For this reason, in this setting we suggest to use the function `cvdglars.fit()` with arguments

```
cvdglars.fit(X, y, family = c("binomial", "poisson"), control = list())
```

where `X` is the design matrix of dimension $N \times p$, `y` is the N-dimensional response vector and `family` is the error distribution used in the model. Like for the `dglars.fit()` function, `control` is a named list of control parameters with the following elements

```
control = list(algorithm = "pc", method = "dgLASSO", nfold = 10,
  foldid = NULL, ng = 100, np = NULL, g0 = NULL, eps = 1.0e-05,
```

```
    nv = NULL, dg_max = 0, nNR = 50, NReps = 1.0e-06, ncrct = 50,
    cf = 0.5, nccd = 1.0e+05)
```

The control parameter `nfold` is used to set the number of folds used for the cross-validation. By default `cvdglars.fit()` uses a ten-fold cross-validation and the prediction error is measured by the deviance. When `nfold` is equal to the sample size, the solution point is estimated by the leave-one-out cross-validation. The optional control parameter `foldid` can be used to pass to the algorithm a N-dimensional vector of non-negative integers used to identify the folds.

From a computational point of view, `cvdglars.fit()` splits the data into `nfold` parts and estimates the dgLARS solution curve using `nfold`-1 parts of the data. To estimate the prediction behavior of the fitted model, the algorithm uses the remaining part of the data and a sequence of `ng` equispaced points of the estimated solution curve to compute the deviance of the model. These steps are repeated `nfold` times and then the mean deviance is computed for each of the `ng` points. The optimal value of the tuning parameter, denoted by $\hat{\gamma}$, is defined as minimum of the mean cross-validation deviance. Finally, using the whole data set, `cvdglars.fit()` runs the adequate subroutine written in Fortran 90 to compute the dgLARS solution curve with control parameter `g0` fixed to $\hat{\gamma}$.

The remaining control parameters of the `cvdglars.fit()` function have the same meaning as those seen in Section 3.1 for the `dglars.fit()` function.

An example of use for a simulated Poisson regression model

To gain more insight about the use of the `cvdglars.fit()` function, we have simulated a data set from a Poisson regression model. Like for the previous example, we assume that only the first two predictors influence the response variable. The corresponding R code is

```
R> set.seed(321)
R> n <- 100
R> p <- 100
R> s <- 2
R> X <- matrix(rnorm(n * p), n, p)
R> bs <- rep(0.5, s)
R> Xs <- X[, 1:s]
R> eta <- drop(0.5 + drop(Xs %*% bs))
R> mu <- poisson()$linkinv(eta)
R> y <- rpois(n, mu)
R> dataset <- data.frame(y = y, X = X)
R> out_cvdglasso_pc <- cvdglars(y ~ ., family = "poisson", data = dataset,
+    control = list(g0 = 0.1, eps = 1.0e-03))
```

Also in this case the values of the control parameters `g0` and `eps` are chosen in order to remove the instability coming from the link function in a high-dimensional setting. The function `cvdglars.fit()` returns an object of class '`cvdglars`'

```
R> class(out_cvdglasso_pc)

[1] "cvdglars"
```

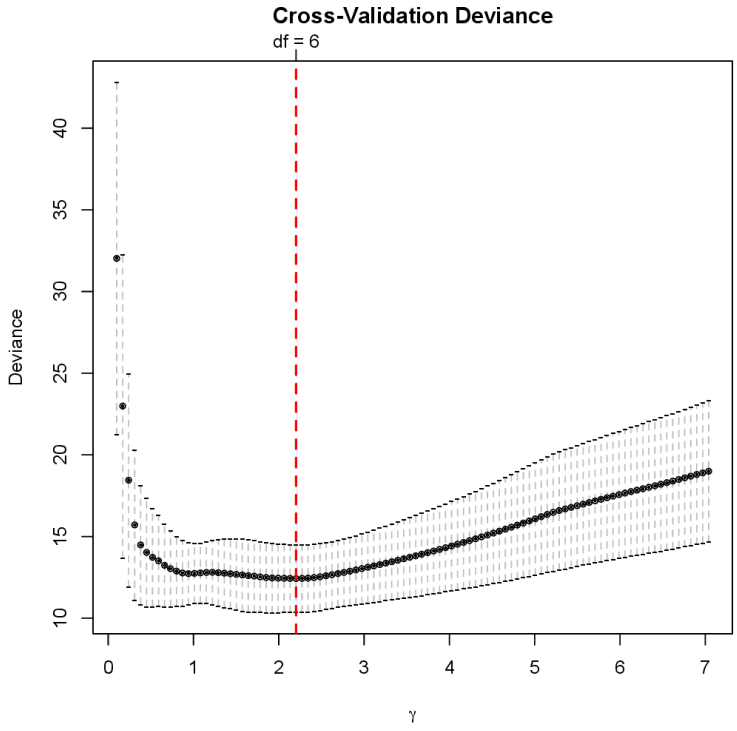

Figure 5: Plot of the '`cvdglars`' object for the simulated data set.

namely a list containing, among the other, the $p + 1$ dimensional vector of the coefficients estimated by k-fold cross-validation, called `beta`, the `ng` dimensional vector of the mean deviance, called `dev_m`, and the estimate of the tuning parameter γ scaled on the interval $[0; 1]$, called `g_hat`.

As for the '`dglars`' object, specific method functions are developed to simplify the study of the output coming from `cvdglars.fit()`. For example, the `print()` and `coef()` methods can be used to print out the basic information stored in the '`cvdglars`' object and to extract the estimated coefficients. Finally, with the `plot()` method for the '`cvdglars`' object it is possible to plot the mean cross-validation deviance as function of the tuning parameter. The optimal value of the tuning parameter is visualized by a vertical red dashed line and the number of estimated non-zero coefficients is also reported. The confidence band for the mean deviance is computed. Figure 5 shows the plot for the results of the model based on our simulated data set. We see that the 10-fold cross-validation method allows us to select a Poisson regression model with only five predictors.

```
R> out_cvdglasso_pc

Call:  cvdglars(formula = y ~ ., family = "poisson", data = dataset,
    control = list(g0 = 0.1, eps = 0.001))

     Int.         X.1         X.2        X.24        X.61        X.78
 0.582782    0.343475    0.249042    0.005463  -0.007751    0.044192
```

```
Cross-validation deviance =  12.43 ( n. fold =  10 )
Algorithm pc ( method = dgLASSO ) with exit = 0
```

As we can see, the two true predictors are included in the estimated optimal solution point.

4. Simulation studies

In this section we compare by simulation studies the behavior of the dgLARS method with the L_1-penalized GLM. To compute the LASSO solution curve we use the R package **glmnet** (Friedman, Hastie, and Tibshirani 2013), which implements the cyclical coordinate descent algorithm proposed in Friedman *et al.* (2010) and the R package **glmpath** (Park and Hastie 2013) which uses the PC algorithm developed by Park and Hastie (2007) to estimate the solution curve.

Our simulation studies are based on a logistic regression model with four scenarios corresponding to four different configurations of the design matrix. The details of the considered scenarios are the following:

(a) $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_p$ sampled from a $N(\mathbf{0}; \Sigma)$, where the diagonal and off-diagonal elements of Σ are 1 and 0, respectively;

(b) the same as (a) but with $\mathsf{COR}(\mathcal{X}_i; \mathcal{X}_j) = \rho^{|i-j|}$; in our study we use $\rho = 0.9$;

(c) the same as (a) with Σ a block diagonal matrix, namely, $\Sigma = \mathrm{diag}(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ where Σ_i is a 10×10 matrix with diagonal elements equal to 1 and off-diagonal elements equal to 0.5;

(d) in this scenario we use as design matrix the gene expression matrix provided by Alon *et al.* (1999). The sample size is equal to 66 and the number of genes p is equal to 2000.

For scenarios (a), (b) and (c) the sample size is equal to one hundred while three different values of p are used, i.e., $p \in (100, 500, 1000)$. In these scenarios only the first five predictors are used to simulate the binary response variable while in scenario (d) the response variable is simulated using five genes randomly selected. The intercept term is equal to one and the non-zero coefficients are equal to two. For each scenario we simulated 500 data sets and a ten-fold cross-validation deviance is used to select the tuning parameter for the L_1-penalized MLE and the parameter γ of the dgLARS method. To make the results comparable, for each data set the same folds were used to determine the value of the tuning parameters of dgLARS/dgLASSO, **glmnet** and **glmpath**, respectively.

In Tables 4 and 5 we report the summary measures used to evaluate the behavior of the considered methods i.e., the total number of variables included in the final model, the false discovery rate, the false positive rate, the false negative rate and the deviance, which was computed using an independent test set.

These results show that the behavior of the dgLARS method is closely related with the covariance structure among the predictors. When we consider scenario (a), dgLARS and dgLASSO exhibit a similar behavior to the L_1-penalized MLE. When the relevant predictors are highly correlated, scenario (b), or when there exists an unknown group structure of the predictors, scenario (c), the results show that the dgLASSO method selects sparser models

Scenario	p		dgLASSO	dgLARS	**glmnet**	**glmpath**
		Size	26.530 (0.586)	28.278 (0.539)	28.216 (0.553)	29.268 (0.538)
		FDR	0.796 (0.006)	0.796 (0.006)	0.798 (0.005)	0.808 (0.005)
	1000	FPR	0.022 (0.001)	0.024 (0.001)	0.024 (0.001)	0.025 (0.001)
		FNR	0.143 (0.011)	0.089 (0.005)	0.103 (0.005)	0.100 (0.005)
		Dev.	87.555 (0.841)	85.023 (0.655)	85.343 (0.595)	84.750 (0.602)
		Size	26.922 (0.549)	27.734 (0.528)	28.450 (0.511)	29.114 (0.534)
		FDR	0.796 (0.006)	0.800 (0.006)	0.810 (0.005)	0.813 (0.005)
(a)	500	FPR	0.046 (0.001)	0.047 (0.001)	0.049 (0.001)	0.050 (0.001)
		FNR	0.144 (0.009)	0.126 (0.007)	0.136 (0.007)	0.140 (0.007)
		Dev.	82.800 (0.673)	81.536 (0.586)	81.666 (0.556)	82.398 (0.619)
		Size	20.704 (0.281)	21.076 (0.291)	21.740 (0.293)	25.374 (0.248)
		FDR	0.732 (0.004)	0.737 (0.004)	0.746 (0.004)	0.793 (0.002)
	100	FPR	0.165 (0.003)	0.169 (0.003)	0.176 (0.003)	0.214 (0.003)
		FNR	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
		Dev.	60.674 (0.510)	60.969 (0.497)	61.582 (0.580)	62.261 (0.614)
		Size	14.650 (0.299)	15.340 (0.302)	23.764 (0.337)	24.940 (0.410)
		FDR	0.656 (0.009)	0.650 (0.008)	0.793 (0.005)	0.802 (0.004)
	1000	FPR	0.011 (0.000)	0.011 (0.000)	0.020 (0.000)	0.021 (0.000)
		FNR	0.185 (0.009)	0.133 (0.006)	0.148 (0.007)	0.144 (0.007)
		Dev.	43.804 (0.903)	41.001 (0.580)	44.400 (0.519)	48.497 (1.263)
		Size	11.258 (0.218)	11.474 (0.224)	20.060 (0.240)	21.364 (0.306)
		FDR	0.625 (0.009)	0.610 (0.009)	0.801 (0.004)	0.809 (0.004)
(b)	500	FPR	0.016 (0.000)	0.016 (0.000)	0.033 (0.000)	0.036 (0.001)
		FNR	0.296 (0.008)	0.258 (0.007)	0.266 (0.006)	0.265 (0.006)
		Dev.	34.622 (0.809)	32.508 (0.499)	34.373 (0.459)	39.663 (0.997)
		Size	8.900 (0.154)	9.182 (0.160)	12.856 (0.183)	15.006 (0.282)
		FDR	0.510 (0.009)	0.502 (0.009)	0.638 (0.006)	0.695 (0.004)
	100	FPR	0.053 (0.002)	0.054 (0.002)	0.091 (0.002)	0.114 (0.003)
		FNR	0.230 (0.008)	0.198 (0.007)	0.167 (0.006)	0.162 (0.006)
		Dev.	41.756 (0.602)	41.151 (0.494)	43.116 (0.558)	44.947 (0.897)

Table 4: Results from the simulation studies (a) and (b); for each scenario we report the mean number of variables included in the final model (Size), the mean false discovery rate (FDR), the mean false positive rate (FPR), the mean false negative rate (FNR), and the mean residual deviance (Dev). Standard errors are in parentheses.

than the L_1-penalized GLM. Also when a design matrix based on real data is used to simulate the response variable (scenario (d)) the dgLASSO method tends to select sparser models.

To end this section, we recall that in general dgLARS is based on a theory completely different from the L_1-penalized MLE implemented in the **glmnet** and **glmpath** packages since dgLARS does not use explicitly a penalized function (see Augugliaro *et al.* 2013, for more details).

Scenario	p		dgLASSO	dgLARS	**glmnet**	**glmpath**
		Size	23.702 (0.421)	24.406 (0.421)	27.656 (0.451)	28.266 (0.478)
		FDR	0.755 (0.006)	0.752 (0.006)	0.787 (0.005)	0.791 (0.005)
	1000	FPR	0.019 (0.000)	0.020 (0.000)	0.023 (0.000)	0.023 (0.000)
		FNR	0.028 (0.006)	0.007 (0.002)	0.014 (0.002)	0.014 (0.002)
		Dev.	56.172 (0.784)	54.353 (0.596)	57.064 (0.552)	58.233 (0.795)
		Size	21.350 (0.356)	21.712 (0.365)	25.210 (0.391)	26.710 (0.472)
		FDR	0.747 (0.005)	0.744 (0.005)	0.782 (0.005)	0.795 (0.005)
(c)	500	FPR	0.034 (0.001)	0.034 (0.001)	0.042 (0.001)	0.045 (0.001)
		FNR	0.071 (0.007)	0.056 (0.005)	0.074 (0.005)	0.071 (0.005)
		Dev.	57.914 (0.715)	56.661 (0.571)	59.130 (0.577)	62.137 (1.270)
		Size	13.078 (0.211)	13.352 (0.224)	15.882 (0.247)	17.910 (0.216)
		FDR	0.568 (0.007)	0.573 (0.008)	0.645 (0.006)	0.701 (0.004)
	100	FPR	0.086 (0.002)	0.089 (0.002)	0.115 (0.003)	0.137 (0.002)
		FNR	0.014 (0.002)	0.013 (0.002)	0.014 (0.002)	0.013 (0.002)
		Dev.	43.294 (0.467)	43.193 (0.465)	45.652 (0.604)	45.664 (0.658)
		Size	11.794 (0.202)	12.344 (0.229)	15.314 (0.260)	15.620 (0.254)
		FDR	0.932 (0.003)	0.932 (0.003)	0.936 (0.002)	0.938 (0.002)
(d)		FPR	0.006 (0.000)	0.006 (0.000)	0.007 (0.000)	0.007 (0.000)
		FNR	0.863 (0.005)	0.854 (0.005)	0.831 (0.005)	0.831 (0.005)
		Dev.	57.175 (0.360)	57.329 (0.347)	57.616 (0.383)	57.604 (0.398)

Table 5: Results from the simulation studies (c) and (d); for each scenario we report the mean number of variables included in the final model (Size), the mean false discovery rate (FDR), the mean false positive rate (FPR), the mean false negative rate (FNR), and the mean residual deviance (Dev). Standard errors are in parentheses.

5. Application to real data sets

5.1. Breast cancer data set

In this section we use the functions available in the **dglars** package to study the sparse structure of a logistic regression model applied to a subset of the breast cancer gene deletion/amplification data set obtained by John Bartlett at the Royal Infirmary, Glasgow (Wit and McClure 2004). The aim of the study is to identify which genes play a crucial role in the severity of the disease, defined as whether or not the patient dies as a result of breast cancer. The data set contains 52 samples, 29 of which are labeled as deceased due to breast cancer. For each sample, 287 gene deletion/amplification measurements are available. A few missing values are imputed using the method proposed by Troyanskaya *et al.* (2001).

To study the considered data set, we first load the data in the R session

```
R> data("breast", package = "dglars")
```

In this example, we estimate the optimal value of the tuning parameter by the 10-fold cross-validation method by using the `cvdglars()` function, i.e.,

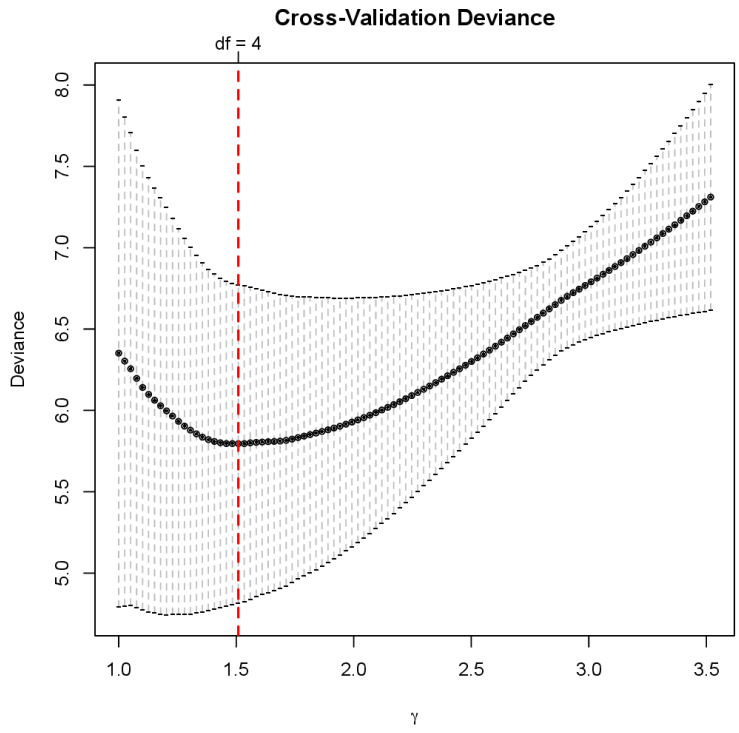

Figure 6: Plot of the 10-fold cross-validation deviance computed for the breast cancer data set. Vertical red dashed line shows that the dgLASSO method finds a logistic regression model with only four non-zero estimated coefficients.

```
R> breast_cvdglasso <- cvdglars(status ~ ., family = "binomial",
+    data = breast, control = list(g0 = 1, eps = 1e-3))
R> breast_cvdglasso

Call:  cvdglars(formula = status ~ ., family = "binomial", data = breast,
    control = list(g0 = 1, eps = 0.001))

           Int.       PTGS2.COX2.         SHGC4.207  WI.2389.D10S1260
         0.2176            0.6565           -4.7278            0.1197

Cross-validation deviance =  5.796 ( n. fold =  10 )
Algorithm pc ( method = dgLASSO ) with exit = 0
```

The plot of the 10-fold cross-validation deviance, shown in Figure 6, has been obtained with this R command

```
R> plot(breast_dglasso)
```

The `print()` output shows that the dgLASSO method selects a logistic regression model with a high level of sparsity since only three genes are found to influence the response variable. Even if the model selected by the 10-fold cross-validation deviance has few variables, we try

to see if it is possible to further reduce the number of genes included in the active set. Then, we first use the function `dglars()`, with design matrix defined using only the genes previously identified, and then we analyze the obtained sequence of logistic regression models with the `summary()` method. Since the MLE of the logistic regression model specified by the subset of identified genes can be computed, we set the argument `complexity = "gdf"` to use the estimator of the generalized degrees of freedom previously described. Using the formula of the model estimated by 10-fold cross-validation deviance, i.e.,

```
R> breast_cvdglasso$formula_cv

status ~ PTGS2.COX2. + SHGC4.207 + WI.2389.D10S1260
```

the following R code confirms the subset of genes previously selected with the `cvdglars()` function:

```
R> breast_dglasso <- dglars(breast_cvdglasso$formula_cv, family = "binomial",
+   data = breast, control = list(g0 = 0))
R> summary(breast_dglasso, k = "BIC", complexity = "gdf")

Call:  dglars(formula = breast_cvdglasso$formula_cv, family = "binomial",
    data = breast, control = list(g0 = 0))

            Sequence          g    Dev     gdf    BIC   Rank
                      3.519e+00  71.39  0.5876  73.72  13
          +SHGC4.207
                      2.755e+00  64.51  0.7323  67.40  12
                      2.329e+00  60.17  0.8116  63.37  11
                      2.168e+00  58.78  0.8499  62.14   8
                      2.103e+00  58.27  0.8664  61.69   7
                      2.072e+00  58.04  0.8743  61.49   5
                      2.057e+00  57.92  0.8782  61.39   4
                      2.049e+00  57.87  0.8801  61.35   3
                      2.042e+00  57.82  0.8820  61.30   2
    +WI.2389.D10S1260
                      1.975e+00  57.18  1.4328  62.84  10
                      1.974e+00  57.17  1.4332  62.83   9
         +PTGS2.COX2.
                      3.857e-02  45.81  3.9120  61.27   1 <-
                      5.350e-09  45.81  4.0000  61.61   6

================================================

Best model identified by BIC criterion ( k = 3.951244 and complexity = gdf ):

 status ~ PTGS2.COX2. + SHGC4.207 + WI.2389.D10S1260

Coefficients:
```

```
          Int.       PTGS2.COX2.          SHGC4.207  WI.2389.D10S1260
       0.01191           3.29593           -9.89851           0.61014


 BIC :  61.27

===

Algorithm pc ( method = dgLASSO ) with exit = 0
```

Then, we can say that the genes selected by the dgLARS method are PTGS2(COX-2), WI.2389.D10S1260 and SHGC4-207. Interestingly, it is known that the expression of COX-2 is upregulated in many cancers. It has been reported in Menter, Schilsky, and Dubois (2010) that inhibiting COX-2 may have benefit in the prevention and treatment of these types of cancer. About SHGC4-207 and WI-2389-D10S1260 much less is known and they could be interesting candidates for further follow-up.

5.2. Duke breast cancer data set

This data set contains 46 of the original 49 patients with primary breast cancer that compose the Duke breast cancer data set studied in West *et al.* (2001). The considered samples can be classified in two groups of the same size, namely the group with estrogen receptor-positive (ER+) and the group with estrogen receptor-negative (ER−). For each sample the gene expression measure of 7129 genes is measured by human HuGeneFL microarray. Aim of this study is to identify a small subset of genes that can be used as prognostic or predictive markers.

As done for the analysis of the previous data set, we first load the data in the R session and then we estimate the optimal value of the tuning parameter by means of the 10-fold cross validation, i.e.,

```
R> data("duke", package = "dglars")
R> duke_cvdglasso <- cvdglars(y ~ ., family = "binomial", data = duke,
+    control = list(g0 = 0.01))
```

The value of the control argument `g0` was chosen after a preliminary study. By the R code

```
R> plot(duke_dglasso)
```

we can see the plot of the 10-fold cross-validation deviance (Figure 7) which shows that the dgLASSO method estimates a logistic regression model with ten parameters, i.e., the intercept term plus the regression coefficients corresponding to a subset of nine genes. To map each manufacturer ID with the corresponding gene name we used the R package **hu6800.db** (Carlson 2013). However, for three genes this mapping is not available. Table 6 shows the basic information about the remaining six identified genes.

The identified set contains several genes that play a central role in breast caner. In a recent study Kristiansen *et al.* (2003) show that CD24 is a new prognostic marker in breast cancer.

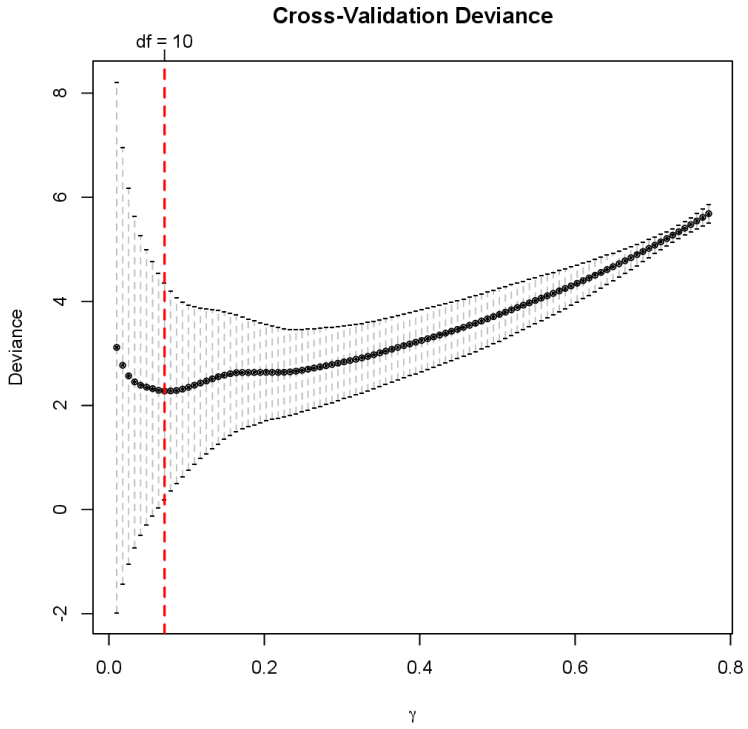

Figure 7: Plot of the 10-fold cross-validation deviance computed for the Duke breast cancer data set.

Manufacturer ID	Symbol	Name
J00116_s_at	COL2A1	Collagen, type II, alpha 1
L22524_s_at	MMP-7	Matrix metallopeptidase 7 (matrilysin, uterine)
L33930_s_at	CD24	CD24 molecule
V00572_at	PGK1	Phosphoglycerate kinase 1
X03635_at	ESR1	Estrogen receptor 1
X17059_s_at	NAT1	N-acetyltransferase 1 (arylamine N-acetyltransferase)

Table 6: Subset of genes identified by the 10-fold cross-validation deviance.

The authors confirm the results previously obtained in Fogel *et al.* (1999) who were the first to describe CD24 expression in breast cancer in an immunohistochemistry study. Previous works have focused on the role of MMP-7 in breast cancer (Bertram and Hass 2008; Bucan, Reimers, Choi, Eddy, and Vogt 2010), while the over-expression of PGK1 has been observed in breast and pancreatic carcinoma and multi-drug resistant ovarian cancer (Duan *et al.* 2002; Zhang *et al.* 2005). Several microarray and proteomic studies indicate that NAT1 is highly expressed in estrogen receptor positive breast cancer (Wakefield *et al.* 2008). Human NAT1 is also a target of cisplatin, one of the most important chemotherapeutic compounds used in the management of various human malignancies (Ragunathan *et al.* 2009). Finally, in recent years ESR1 is shown to be frequently expressed in breast cancer (Holst *et al.* 2007). Several studies showed statistically significant associations between ESR1 polymorphisms and breast cancer, see for example Gold *et al.* (2004) and Wang *et al.* (2007).

As a consequence of the complete data separation, the MLE of the logistic regression model specified using only the subset of genes previously identified cannot be computed (Albert and Anderson 1984), then we use the BIC, with complexity defined by the number of non-zero estimated coefficients, to try to see if it is possible to identify a smaller subset. For this reason, we first estimate the dgLASSO solution curve by the code

```
R> duke_dglasso <- dglars(duke_cvdglasso$formula_cv, family = "binomial",
+    data = duke)
```

and then, from the `summary()` output

```
R> summary(duke_dglasso)
```

```
Call:  dglars(formula = duke_cvdglasso$formula_cv, family = "binomial",
    data = duke)

               Sequence          g        Dev  df    BIC   Rank
                         7.719e-01  6.377e+01   0  63.77  22
             +X03635_at
                         2.569e-01  2.740e+01   2  35.06   2
                         2.545e-01  2.725e+01   2  34.91   1 <-
             +M34516_at
                         2.393e-01  2.623e+01   3  37.72   4
                         2.392e-01  2.623e+01   3  37.72   3
           +X17059_s_at
                         2.059e-01  2.368e+01   4  38.99  10
                         2.054e-01  2.364e+01   4  38.96   9
           +L22524_s_at
                         2.042e-01  2.354e+01   5  42.69  16
             +M59815_at
                         1.901e-01  2.220e+01   6  45.17  18
                         1.900e-01  2.218e+01   6  45.15  17
    +HG3431.HT3616_s_at
                         1.870e-01  2.184e+01   7  48.64  21
                         1.855e-01  2.167e+01   7  48.47  20
                         1.840e-01  2.150e+01   7  48.30  19
           +J00116_s_at
                         1.033e-01  9.476e+00   8  40.11  13
                         1.014e-01  9.221e+00   8  39.85  11
             +V00572_at
                         7.994e-02  6.265e+00   9  40.72  15
                         7.858e-02  6.091e+00   9  40.55  14
           +L33930_s_at
                         3.934e-02  1.805e+00  10  40.09  12
                         1.972e-02  4.954e-01  10  38.78   8
                         9.911e-03  1.311e-01  10  38.42   7
                         5.005e-03  3.435e-02  10  38.32   6
                         9.997e-05  1.458e-05  10  38.29   5
```

```
============================================

Best model identified by BIC criterion ( k = 3.828641 and complexity = df ):

 y ~ X03635_at

Coefficients:

     Int.  X03635_at
  -12.269      1.353


 BIC :  34.91

===

Algorithm pc ( method = dgLASSO ) with exit = 0
```

we see that we can identify a logistic regression model specified using only ESR1 as predictor.

6. Summary

In this paper we have described the R package **dglars**. This package implements the differential geometric extension of the LARS method proposed in Augugliaro *et al.* (2013) and called dgLARS. The core of the package are two functions implementing a predictor-corrector algorithm and a cyclic coordinate descent algorithm to compute the dgLARS solution curve. In order to implement these two algorithms in an efficient way, the main code is written in Fortran 90 and handled in R by specific wrapper functions. This package also provides a function to estimate the generalized degrees of freedom proposed in Augugliaro *et al.* (2013). When we have the MLEs of the GLM specified by all the considered predictors, it is possible to use a more accurate measure of goodness of fit than the number of non-zero estimated coefficients to select the optimal points of the solution curve. The use of these functions is shown by means of simulated data sets and application to real data sets. The output of the functions are presented in a way that is easy to interpret for people familiar with standard `lm()`, `glm()` or `gam()` output.

The package **dglars** is available under the general public license (GPL $\geq$ 2) from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=dglars`. In future versions of the package we are going to implement other distribution families such as gamma and inverse Gaussian distributions.

References

Akaike H (1973). "Information Theory as an Extension of the Maximum Likelihood Principle." In BN Petrov, F Czaki (eds.), *Second International Symposium on Information Theory*, pp. 267–281. Akademiai Kiado, Budapest.

Albert A, Anderson JA (1984). "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika*, **71**(1), 1–10.

Allgower E, Georg KM (2003). *Introduction to Numerical Continuation Methods*. SIAM, New York.

Alon U, Barkai N, Notterman D, Gish K, Ybarra S, Mack D, Levine A (1999). "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays." *Proceedings of the National Accademy of Sciences of the United States of America*, **96**(12), 6745–6750.

Augugliaro L (2014). ***dglars**: Differential Geometric LARS (dgLARS) Method*. R package version 1.0.5, URL http://CRAN.R-project.org/package=dglars.

Augugliaro L, Mineo AM, Wit EC (2012). "Differential Geometric LARS via Cyclic Coordinate Descent Method." In *COMPSTAT 2012 – Proceedings in Computational Statistics*, pp. 67–79. Limassol, Cyprus.

Augugliaro L, Mineo AM, Wit EC (2013). "Differential Geometric Least Angle Regression: A Differential Geometric Approach to Sparse Generalized Linear Models." *Journal of the Royal Statistical Society B*, **75**(3), 471–498.

Bertram C, Hass R (2008). "MMP-7 Is Involved in the Aging of Primary Human Mammary Epithelial Cells (HMEC)." *Experimental Gerontology*, **43**(3), 209–217.

Bucan V, Reimers K, Choi CY, Eddy MT, Vogt PM (2010). "The Anti-Apoptotic Protein Lifeguard Is Expressed in Breast Cancer Cells and Tissues." *Cellular & Molecular Biology Letters*, **15**(2), 296–310.

Candes E, Tao T (2007). "The Dantzig Selector: Statistical Estimation When p Is much Larger than n." *The Annals of Statistics*, **35**(6), 2313–2351.

Carlson M (2013). ***hu6800.db**: Affymetrix HuGeneFL Genome Array Annotation Data (Chip hu6800)*. R package version 2.14.0, URL http://www.bioconductor.org/packages/release/data/annotation/html/hu6800.db.html.

Duan Z, Lamendola DE, Yusuf RZ, Penson RT, Preffer FI, Seiden MV (2002). "Overexpression of Human Phosphoglycerate Kinase 1 (PGK1) Induces a Multidrug Resistance Phenotype." *Anticancer Research*, **22**(4), 1933–1941.

Efron B (1986). "How Biased Is the Apparent Error Rate of a Prediction Rule?" *Journal of the American Statistical Association*, **81**(394), 461–470.

Efron B (2004). "The Estimation of Prediction Error: Covariance Penalties and Cross-Validation." *Journal of the American Statistical Association*, **99**(467), 619–632.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression." *The Annals of Statistics*, **32**(2), 407–499.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360.

Fan J, Lv J (2010). "A Selective Overview of Variable Selection in High Dimensional Feature Space." *Statistica Sinica*, **20**(1), 101–148.

Fogel M, Friederichs J, Zeller Y, Husar M, Smirnov A, Roitman L, Altevogt P, Sthoeger ZM (1999). "CD24 Is a Marker for Human Breast Carcinoma." *Cancer Letters*, **143**(1), 87–94.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. URL http://www.jstatsoft.org/v33/i01/.

Friedman J, Hastie T, Tibshirani R (2013). ***glmnet**: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 1.9-5, URL http://CRAN.R-project.org/package=glmnet.

Gold B, Kalush F, Bergeron J, Scott K, Mitra N, Wilson K, Ellis N, Huang H, Chen M, Lippert R, Halldorsson BV, Woodworth B, White T, Clark AG, Parl FF, Broder S, Dean M, Offit K (2004). "Estrogen Receptor Genotypes and Haplotypes Associated with Breast Cancer Risk." *Cancer Research*, **64**(24), 8891–8900.

Hastie T, Tibshirani R, Friedman JH (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer-Verlag, New York.

Holst F, Stahl PR, Ruiz C, Hellwinkel O, Jehan Z, Wendland M, Lebeau A, Terracciano L, Al-Kuraya K, Jänicke F, Sauter G, Simon R (2007). "Estrogen Receptor Alpha (ESR1) Gene Amplification Is Frequent in Breast Cancer." *Nature Genetics*, **39**(5), 655–660.

James GM, Radchenko P (2009). "A Generalized Dantzig Selector with Shrinkage Tuning." *Biometrika*, **96**(2), 323–337.

Kristiansen G, Winzer KJ, Mayordomo E, Bellach J, Schlüns K, Denkert C, Dahl E, Pilarsky C, Altevogt P, Guski H, Dietel M (2003). "CD24 Expression Is a New Prognostic Marker in Breast Cancer." *Clinical Cancer Research*, **9**(13), 4906–4913.

Madigan D, Ridgeway G (2004). "Discussion to Least Angle Regression." *The Annals of Statistics*, **32**(2), 465–469.

Menter DG, Schilsky RL, Dubois RN (2010). "Cyclooxygenase-2 and Cancer Treatment: Understanding the Risk Should Be Worth the Reward." *Clinical Cancer Research*, **16**(5), 1384–1390.

Park MY, Hastie T (2007). "L_1-Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society B*, **69**(4), 659–677.

Park MY, Hastie T (2013). ***glmpath**: L1 Regularization Path for Generalized Linear Models and Cox Proportional Hazards Model*. R package version 0.97, URL http://CRAN.R-project.org/package=glmpath.

Ragunathan N, Dairou J, Pluvinage B, Martins M, Dupret JM, Rodrigues-Lima F (2009). "Human Arylamine N-Acetyltransferase 1 (NAT1) as a Target of Chemotherapeutic Drugs in Breast Cancer: Cisplatin as a Model." *Molecular and Cellular Pharmacology*, **1**(1), 7–10.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001). "Missing Value Estimation Methods for DNA." *Bioinformatics*, **17**(6), 520–525.

Wakefield L, Robinson J, Long H, Ibbitt JC, Cooke S, Hurst HC, Sim E (2008). "Arylamine N-Acetyltransferase 1 Expression in Breast Cancer Cell Lines: A Potential Marker in Estrogen Receptor-Positive Tumors." *Genes, Chromosomes & Cancer*, **47**(2), 118–126.

Wang J, Higuchi R, Modugno F, Li J, Umblas N, Lee J, Lui LY, Ziv E, Tice JA, Cummings SR, Rhees B (2007). "Estrogen Receptor Alpha Haplotypes and Breast Cancer Risk in Older Caucasian Women." *Breast Cancer Research and Treatment*, **106**(2), 273–280.

West M, Blanchette C, Dressman H, Huang E, Ishida S, Spang R, Zuzan H, Olson JA, Marks JR, Nevins JR (2001). "Predicting the Clinical Status of Human Breast Cancer by Using Gene Expression Profiles." *Proceedings of the National Accademy of Sciences of the United States of America*, **98**(20), 11462–11467.

Wit EC, McClure JD (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Chichester.

Zhang CH (2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty." *The Annals of Statistics*, **38**(2), 894–942.

Zhang D, Tai LK, Wong LL, Chiu LL, Sethi SK, Koay ESC (2005). "Proteomic Study Reveals that Proteins Involved in Metabolic and Detoxification Pathways Are Highly Expressed in HER-2/Neu-Positive Breast Cancer." *Molecular & Cellular Proteomics*, **4**(11), 1686–1696.

Zou H, Hastie T, Tibshirani R (2007). "On the Degrees of Freedom of the Lasso." *The Annals of Statistics*, **35**(5), 2173–2192.

Affiliation:

Luigi Augugliaro, Angelo M. Mineo
Dipartimento di Scienze Economiche, Aziendali e Statistiche
University of Palermo
90128 Palermo, Italy
Telephone: +39/091/238-95242, +39/091/238-95300
Fax: +39/091/485726
E-mail: luigi.augugliaro@unipa.it, angelo.mineo@unipa.it
URL: http://dssm.unipa.it/augugliaro/, http://dssm.unipa.it/elio/

Ernst C. Wit
Institute of Mathematics and Computing Science, IWI
University of Groningen, RUG
Bernoulliborg, Rm. 446, Nijenborgh 9
9747 AG Groningen, The Netherlands
Telephone: +31/50/3635170
Fax: +31/50/363380
E-mail: e.c.wit@rug.nl
URL: http://www.math.rug.nl/~ernst/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 59, Issue 8 *Submitted:* 2012-11-13
August 2014 *Accepted:* 2014-05-05