Noname manuscript No.
(will be inserted by the editor)

Group linear algorithm with sparse principal decomposition

A variable selection and clustering method for generalized linear models

Juan C. Laria · M. Carmen Aguilera-Morillo · Rosa E. Lillo

Received: date / Accepted: date

Abstract This paper introduces the Group Linear Algorithm with Sparse Principal decomposition, an algorithm for supervised variable selection and clustering. Our approach extends the Sparse Group Lasso regularization to calculate clusters as part of the model fit. Therefore, unlike Sparse Group Lasso, our idea does not require prior specification of clusters between variables. To determine the clusters, we solve a particular case of sparse Singular Value Decomposition, with a regularization term that follows naturally from the Group Lasso penalty. Moreover, this paper proposes a unified implementation to deal with, but not limited to, linear regression, logistic regression, and proportional hazards models with right-censoring. Our methodology is evaluated using both biological and simulated data, and details of the implementation in R and hyperparameter search are discussed.

Keywords Regression · Classification · Feature clustering · Statistical computing

1 Introduction

In recent years, penalized regression problems for variable selection have become very popular. Since the introduction of Lasso (Tibshirani, 1996) as a

Juan C. Laria
TomTom Maps-Analytics, Madrid, Spain
E-mail: juank.laria@gmail.com

M. Carmen Aguilera-Morillo
Department of Applied Statistics and Operational Research and Quality, Universitat Politècnica de València, Spain
UC3M-BS Santander Big Data Institute, Getafe, Spain

Rosa E. Lillo
Department of Statistics, University Carlos III of Madrid
UC3M-BS Santander Big Data Institute, Getafe, Spain

regularization term for linear models, many extensions have dealt with variable selection by penalizing the loss function. Most of these extensions are limited to linear regression, but some of them, such as Elastic-Net (Zou and Hastie, 2005), Group Lasso (Zhou and Zhu, 2010) or recently Sparse Group Lasso (Simon et al., 2013) have been also extended to generalized linear models (GLMs).

In this paper, we focus on the Sparse Group Lasso as a variable selection method in high-dimensional problems. The hypothesis of the existence of previously known clusters among the variables poses a significant practical difficulty for this method to be applied to every supervised problem. Besides, the Sparse Group Lasso penalty function is the linear combination of a Lasso penalty (ℓ_1 norm) and a Group Lasso penalty (ℓ_2 norm), so there are at least two regularization hyperparameters (plus one for each group, usually fixed). In most of the applications of the Sparse Group Lasso, the parameters are either fixed based on a prior information about the data, or chosen to minimize some error function in a grid of possible values. In that sense, Laria et al. (2019) proposed a gradient-free coordinate descent algorithm, which allows the automatic selection of the regularization parameters in the SGL. However, the problem of grouping the variables was not solved. In genetic or financial applications, there is a growing demand not only for building predictive models but also for clustering the variables. Recent papers highlight the importance of group structures in variable selection (Luo and Chen, 2020; Ciuperca, 2020; Zhang et al., 2020).

The main methodological contribution of this article is the formal definition of GLASP, a Group Linear Algorithm with Sparse Principal decomposition. GLASP is an extension of the Sparse Group Lasso, that, not only avoids the need for a specification of clusters among the variables, but also computes such clusters during the model fitting process. Therefore, apart from a predictive model, GLASP can be considered as a supervised variable clustering algorithm.

The GLASP specification is motivated by the Cluster Elastic Net (CEN) (Witten et al., 2014), where the authors extend the elastic-net to obtain groups between variables using k-means, besides variable selection and model fitting. Recently, some extensions have considered multivariate response CEN, for example, Price and Sherwood (2017) and Ren et al. (2020). A minor disadvantage of CEN is that the number of clusters between variables has to be specified initially. Unlike CEN, our GLASP algorithm can obtain a smaller number of groups than initially specified.

From an algorithmic point of view, GLASP has two parts. The first is an accelerated block gradient descent algorithm to adjust the Sparse Group Lasso with an arbitrary and flexible error function, which is a linear combination of the model loss function and a differentiable regularization term. The second is a particular type of regularized Singular Value Decomposition, with a penalty function adapted to this specific problem, in order to find the groups.

The method proposed in this paper is, to the best of our knowledge, the first extension of the Sparse Group Lasso that computes groups automatically. The internal supervised variable clustering algorithm is also an original contribution and integrates naturally within the Group Lasso penalty. Moreover, our implementation provides the flexibility to change the risk function and address any regression problem.

This paper is organized as follows. Section 2 introduces GLASP as the solution to a problem involving sparsity, clustering, and structure assumptions on the variables. Section 3 describes in detail the solutions of both sub-problems addressed, with particular emphasis on the internal optimization algorithms. Later, Section 4 compares our approach with other linear regression methods that perform variable selection and clustering. Although a general notation is adopted from the beginning to refer to the loss function, the main differences when a linear, logistic or Cox survival model with right-censoring is adjusted with GLASP are explained in Section 5. For the latter, additional details related to prediction are presented, as well as a simulation study on survival data. Moreover, relevant details and practical examples related to the implementation of GLASP in R language are illustrated in Section 6, with special emphasis on its tidy interface, and the optimization of hyper-parameters. Section 7 illustrates an application of GLASP to gene clustering and survival prediction with right-censored data. Finally, Section 8 discusses the implications of our work and future directions of research.

2 Formulation of GLASP

Under the penalized general linear regression framework, we have a data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times p}$, a response vector $\boldsymbol{y} \in \mathbb{R}^{N \times 1}$, and we are interested in finding $\boldsymbol{\beta} \in \mathbb{R}^{p \times 1}$ such that $L(\boldsymbol{\beta}) + \phi(\boldsymbol{\beta})$ is minimum. Here $\phi : \mathbb{R}^{p \times 1} \to \mathbb{R}^{+}$ is some penalty, and $L : \mathbb{R}^{p \times 1} \to \mathbb{R}$ is an empirical risk function that measures how good can we approximate $\boldsymbol{y}$ knowing $\boldsymbol{X\beta}$. Throughout this paper, and without loss of generality, we will assume that the data matrix $\boldsymbol{X}$ is standardized to have mean 0 and variance 1 in each column (i.e. $\bar{\boldsymbol{X}}_j = 0$ and $\boldsymbol{X}_j^\top \boldsymbol{X}_j = 1$ for every $j = 1, 2 \ldots p$). This is important for the computations in next sections. We will make the following extra assumptions:

1. (Sparsity) There is a small number of columns of $\boldsymbol{X}$ that are actually related to $\boldsymbol{y}$, and therefore many components of $\boldsymbol{\beta}$ are exactly zero.
2. (Clustering) There is a (possible unknown) number K of unknown groups, or clusters, among the variables of $\boldsymbol{X}$.
3. (Structure) For every group, there is associated a latent variable that summarizes the information provided by all the variables in that cluster. In linear models, information is measured in terms of linear predictors. A variable $\boldsymbol{X}_j \in \mathbb{R}^{N \times 1}, j = 1, 2 \ldots p$, provides information to the model through $\boldsymbol{X}_j \boldsymbol{\beta}_j$. Knowing the true groups will improve the estimation of $\boldsymbol{\beta}$, and knowing $\boldsymbol{\beta}$ will give us insight into the groups.

These assumptions are aligned with those of Witten et al. (2014). However, we want to remark that often, the number K is unknown. In addition, we do not want to assume beforehand that $\boldsymbol{X}_j \beta_j$ and $\boldsymbol{X}_l \beta_l$ are close in the squared euclidean distance, for $\boldsymbol{X}_j$ and $\boldsymbol{X}_l$ in the same group.

Solving the sparse regression problem and, at the same time, finding the clusters in the columns of $\boldsymbol{X}$, motivates the GLASP optimization problem,

$$\min_{\boldsymbol{\beta},\boldsymbol{W},\boldsymbol{T}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 + \frac{\lambda_3}{2} \left\|\boldsymbol{\mathcal{X}} - \boldsymbol{T}\boldsymbol{W}^\top\right\|_F^2 \right\}, \tag{1}$$

where

- $\|\cdot\|_F^2$ is the squared Frobenius norm, given by $\|\boldsymbol{M}\|_F^2 = Tr(\boldsymbol{M}\boldsymbol{M}^\top)$.
- $\boldsymbol{W} \in \mathbb{R}^{p\times K}$ is an orthogonal matrix with cluster information, $\boldsymbol{W}^\top\boldsymbol{W}$ diagonal.
- $\boldsymbol{T} \in \mathbb{R}^{N\times K}$ (latent groups) is a low-rank unitary representation of the linear predictors, $\boldsymbol{T}^\top\boldsymbol{T} = \boldsymbol{I}_K$.
- $\boldsymbol{\mathcal{X}} = \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top)\beta_j \in \mathbb{R}^{N\times p}$ is the matrix of linear predictors, where e_j is the j–th unit vector in the canonical basis of $\mathbb{R}^{p\times 1}$.
- $\boldsymbol{J}_k = \sqrt{\|\boldsymbol{W}_k\|_0} \sum_{j=1}^{p} (e_j e_j^\top)\mathbb{1}(W_{jk} \neq 0)$ is a diagonal projection matrix such that $\|\boldsymbol{J}_k\boldsymbol{\beta}\|_2$ is the euclidean norm of the vector of coefficients associated with group k, penalized by the size of the group. Here $\|\boldsymbol{W}_k\|_0$ denotes the number of elements in column k–th of $\boldsymbol{W}$ that are non-zero, which is the size of group k.
- $\lambda_1, \lambda_2, \lambda_3$ are regularization hyperparameters.

Problem (1) is a non-convex optimization problem, and finding the global optimum would require to search for orthogonal matrices $\boldsymbol{T}$, rotation matrices $\boldsymbol{W}$ and coefficient vectors $\boldsymbol{\beta}$ that minimize (1). This is impractical, and we propose a two-step iterative approach to find a local minimum of (1). See, for example Witten et al. (2014). If we minimize (1) only with respect to $\boldsymbol{\beta}$, we obtain the sub-problem

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 + \frac{\lambda_3}{2} \left\|\boldsymbol{\mathcal{X}} - \boldsymbol{T}\boldsymbol{W}^\top\right\|_F^2 \right\}, \tag{2}$$

On the other hand, if we let $\boldsymbol{\beta}$ fixed and minimize (1) with respect to $\boldsymbol{W}$ and T, then (1) becomes,

$$\min_{\boldsymbol{W},\boldsymbol{T}} \left\{ \frac{\lambda_3}{2} \left\|\boldsymbol{\mathcal{X}} - \boldsymbol{T}\boldsymbol{W}^\top\right\|_F^2 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 \right\}, \tag{3}$$

Our algorithmic methodology is based on minimizing (2) and (3) until convergence.

Remark 1 If $\lambda_3 = 0$, problem (2) is the Sparse Group Lasso.

In this case, (2) becomes,

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{k=1}^{K} \|\boldsymbol{J}_k \boldsymbol{\beta}\|_2 \right\}$$

By construction, $\boldsymbol{J}_k$ is a projection matrix such that $\|\boldsymbol{J}_k\boldsymbol{\beta}\|_2 = \sqrt{\|\boldsymbol{W}_k\|_0}\,\left\|\boldsymbol{\beta}^{(k)}\right\|_2$, where $\boldsymbol{\beta}^{(k)}$ are the coefficients in group k. The previous expression is equivalent to the Sparse Group Lasso (Simon et al., 2013).

Remark 2 In general, for $\boldsymbol{T}, \boldsymbol{W}$ fixed, the penalization $\varphi(\boldsymbol{\beta}) = \lambda_3/2 \left\|\boldsymbol{\mathcal{X}} - \boldsymbol{T}\boldsymbol{W}^\top\right\|_F^2$ does not shrink $\boldsymbol{\beta}$ towards zero, and therefore, the regularization function in (2) may not shrink to zero, but to some other vector.

After some algebra, $\varphi(\boldsymbol{\beta})$ can be written in the form

$$\varphi(\boldsymbol{\beta}) = (\beta_1 - c_1)^2/a_1 + (\beta_2 - c_2)^2/a_2 + \cdots + (\beta_p - c_p)^2/a_p - r^2,$$

where a_j, c_j, r are values depending on $\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{T}$. The contour levels of $\varphi(\boldsymbol{\beta})$ correspond to ellipsoids in $\mathbb{R}^p$, centered at $(c_1, \ldots c_p)$. To see this, we will plot the penalty as a function of $\boldsymbol{\beta}$. As a toy example, consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{100\times 3}$, with $N(0,1)$ columns, such that $cov(X_1, X_2) = 0$, $cov(X_2, X_3) = 0$ and $cov(X_1, X_3) = 0.5$. Let $\tilde{\boldsymbol{c}} = (0.5, 0.25, 0.1)^\top$, and $\boldsymbol{U\Sigma V}^\top$ the singular value decomposition of $\boldsymbol{X}\tilde{\boldsymbol{c}}$. We choose $\boldsymbol{T} = \boldsymbol{U}$ and $\boldsymbol{W}$ such that $\boldsymbol{W}^\top\boldsymbol{W}$ diagonal, but close to $\boldsymbol{V\Sigma}$. Then, we have $\boldsymbol{T}^\top\boldsymbol{T} = \boldsymbol{I}$ and $\boldsymbol{W}$ is approximately given by,

$$\boldsymbol{W} = \begin{pmatrix} 4.97 & 0 \\ 0 & 2.46 \\ 0.37 & 0 \end{pmatrix},$$

such that β_1 and β_2 are in different groups. The contour plot for the GLASP penalty is shown in Figure 1a, compared with Lasso (Figure 1b), Sparse Group Lasso (Figure 1c), and GLASP with $\lambda_1 = \lambda_2 = 0$ (Figure 1d).

Remark 3 For a fixed $\boldsymbol{\beta}$, problem (3) can be written in the form

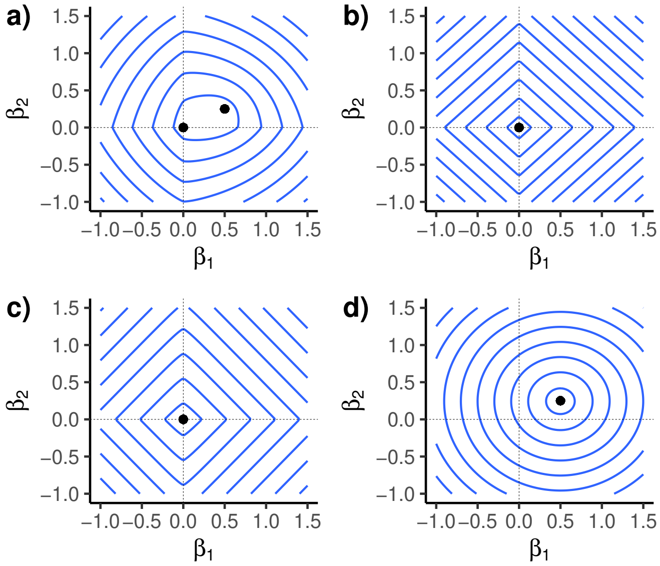
$$\min_{\boldsymbol{W},\boldsymbol{T}} \left\{ \left\|\boldsymbol{\mathcal{X}} - \boldsymbol{T}\boldsymbol{W}^\top\right\|_F^2 + \gamma P(\boldsymbol{W}) \right\}, \tag{4}$$

where $\gamma = 2\lambda_2/\lambda_3$ and

$$P(\boldsymbol{W}) = \sum_{k=1}^{K} \left(\sum_{j=1}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{W}_{jk} \neq 0)\, \|\boldsymbol{W}_k\|_0 \right)^{1/2}. \tag{5}$$

This is a penalized low-rank approximation problem, and in this case, P is a sparsity penalty.

Fig. 1 Contour plots for the GLASP(a), the Lasso (b), the Sparse Group Lasso (c) and GLASP with $\lambda_1 = \lambda_2 = 0$ (d).



This problem written in the general form (4) is very similar to those investigated by Shen and Huang (2008). The first part is a low rank approximation problem, which is known to be solved by the singular value decomposition. In our case, the challenging part is the function P, which is non-differentiable and non-convex. However, it is clear that P is a sparsity penalty, and therefore, (4) will force sparsity in $\boldsymbol{W}$. We propose a solution based to the *sparse PCA via regularized SVD* of Shen and Huang (2008), but considering our function P as penalty for $\boldsymbol{W}$, instead of common choices.

3 Algorithms

In this section, we detail the computations to solve the GLASP problem, separated into two sub-problems, defined in (2) and (3), respectively.

3.1 Internal optimization by groups

Consider (2) for $\boldsymbol{W}, \boldsymbol{T}$ fixed. The final algorithm is a block gradient descent method. We found this solution to be very fast to solve convex optimization problems in a general context, where there is a differentiable loss and a sub-differentiable penalty. Recent papers dealing with the Sparse Group Lasso and extensions have also adopted similar approaches (Simon et al., 2013; Ren et al., 2020; Laria et al., 2019).

Problem (2) can be minimized using a cyclic group-wise gradient descent. Assume that vector $\boldsymbol{\beta}$ is fixed for all groups but k–th, and without loss of generality, assume that the coefficients in group k are $\beta_1, \beta_2 \ldots \beta_{p_k}$. To avoid difficult notation, throughout this section $\boldsymbol{\beta} = (\beta_1, \beta_2 \ldots \beta_{p_k})^\top$ will denote the coefficient vector for group k and $p_k = \|\boldsymbol{W}_k\|_0$ its number of elements. Since the remaining groups are fixed, and using the definition of the Frobenius norm, (2) becomes

$$\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sqrt{p_k} \|\boldsymbol{\beta}\|_2 + \frac{\lambda_3}{2} \sum_{j=1}^{p_k} \left\| \boldsymbol{X}_j \beta_j - \boldsymbol{T}\boldsymbol{W}_{j\cdot}^\top \right\|_2^2 \right\}. \tag{6}$$

To solve (6) we will use the *fast iterative shrinkage-thresholding algorithm* (FISTA) (Beck and Teboulle, 2009).

Consider the general optimization problem

$$\min_{\boldsymbol{\beta}} \left\{ F(\boldsymbol{\beta}) := R(\boldsymbol{\beta}) + \Phi(\boldsymbol{\beta}) \right\}, \tag{7}$$

where

- $R : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is a smooth convex function, continuously differentiable with Lipschitz continuous gradient ∇R (with Lipschitz constant $\mathcal{L}(R)$), such that
$$\|\nabla R(\boldsymbol{\beta}) - \nabla R(\boldsymbol{\beta}_0)\|_2 \leq \mathcal{L}(R) \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2 .$$
- $\Phi : \mathbb{R}^{p\times 1} \to \mathbb{R}$ is a continuous convex function which is possibly non-smooth.

Consider (6) in the form (7), taking

$$R(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \frac{\lambda_3}{2} \sum_{j=1}^{p_k} \left\| \boldsymbol{X}_j \beta_j - \boldsymbol{T}\boldsymbol{W}_{j\cdot}^\top \right\|_2^2 ,$$

and

$$\Phi(\boldsymbol{\beta}) = \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \sqrt{p_k} \|\boldsymbol{\beta}\|_2 .$$

The core of the FISTA algorithm is to consider, for any $t > 0$, the quadratic approximation of $F(\boldsymbol{\beta})$ at a given point $\boldsymbol{\beta}_0$, given by,

$$M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = R(\boldsymbol{\beta}_0) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \nabla R(\boldsymbol{\beta}_0) + \frac{1}{2t} \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2 + \Phi(\boldsymbol{\beta}), \tag{8}$$

which admits a unique minimizer (that we refer to as *update function*)

$$\begin{aligned} U_t(\boldsymbol{\beta}_0) &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \{ M_t(\boldsymbol{\beta}, \boldsymbol{\beta}_0) \} \\ &= \operatorname{argmin}_{\boldsymbol{\beta}} \left\{ M_t^*(\boldsymbol{\beta}, \boldsymbol{\beta}_0) = \tfrac{1}{2} \|\boldsymbol{\beta} - \boldsymbol{B}_{\mathbf{0}}\|_2^2 + t\Phi(\boldsymbol{\beta}) \right\}, \end{aligned} \tag{9}$$

where $\boldsymbol{B}_{\mathbf{0}} = \boldsymbol{\beta}_0 - t\nabla R(\boldsymbol{\beta}_0)$. The idea of the *iterative shrinkage-thresholding algorithm* (ISTA) algorithm is to produce a descent sequence for F via $\boldsymbol{\beta}_{(k+1)} \leftarrow$

$U_t(\boldsymbol{\beta}_{(k)})$, choosing t carefully such that $t < 1/\mathcal{L}(R)$. If the Lipschitz constant $\mathcal{L}(R)$ is unknown, t_k is found in each step using a backtracking stepsize rule, to be the maximum $t > 0$ such that,

$$F(U_t(\boldsymbol{\beta}_{(k)})) \leq M_t(U_t(\boldsymbol{\beta}_{(k)}), \boldsymbol{\beta}_{(k)}). \tag{10}$$

To accelerate the global rate of convergence from $1/k$ (ISTA) to $1/k^2$, the FISTA algorithm updates $\boldsymbol{\beta}_{(k)}$ according to

$$\boldsymbol{\beta}_{(k+1)} \leftarrow U_{t_k}(\boldsymbol{\beta}_{(k)}) + \frac{l_k - 1}{l_{k+1}}(U_{t_k}(\boldsymbol{\beta}_{(k)}) - U_{t_k-1}(\boldsymbol{\beta}_{(k-1)})), \tag{11}$$

where $l_{k+1} = (1 + \sqrt{1 + 4l_k^2})/2$, $l_1 = 1$.

The most difficult part to formulate in our algorithm to solve (6) using FISTA, is to minimize M_t^* as a function of $\boldsymbol{\beta}$, which in our case is given by,

$$M_t^*(\boldsymbol{\beta}) = \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{B_0}\right\|_2^2 + t\lambda_1 \left\|\boldsymbol{\beta}\right\|_1 + t\lambda_2\sqrt{p_k}\left\|\boldsymbol{\beta}\right\|_2 \tag{12}$$

Next proposition provides the update function (9) corresponding to M_t^* in (12).

Proposition 1 *The update function of problem* (6) *is given by,*

$$U_t(\boldsymbol{\beta}_0) = \left(1 - \frac{t\lambda_2\sqrt{p_k}}{\left\|S(\boldsymbol{B_0}, t\lambda_1)\right\|_2}\right)_+ S(\boldsymbol{B_0}, t\lambda_1),$$

where S is the coordinate-wise soft threshold operator,

$$S(\boldsymbol{z}, \lambda)_i = sign(z_i)(|z_i| - \lambda)_+$$

The following proposition provides conditions for $\boldsymbol{\beta} = \mathbf{0}$ to be the minimizer of (6). If we know, after a simple computation, that $\boldsymbol{\beta} = \mathbf{0}$, then we can skip the FISTA optimization for the coefficients in that group. Moreover, these conditions are also upper bounds for the maximum values of the hyperparameters, such that $\boldsymbol{\beta} \neq 0$.

Proposition 2 $\boldsymbol{\beta} = \mathbf{0}$ *is the minimizer of* (6) *if*

$$\left\|S(\nabla R(\mathbf{0}), \lambda_1)\right\|_2 \leq \lambda_2\sqrt{p_k}. \tag{13}$$

In particular, it is also true if,

$$\max_j |\nabla_j R(\mathbf{0})| \leq \lambda_1. \tag{14}$$

The proofs of Propositions 1 and 2 can be found in the Appendix.

3.2 Group optimization

This section describes the solution that we propose for sub-problem (3). In addition, we will assume that there are no overlapping groups.

As stated in Remark 3, when $\boldsymbol{\beta}$ is fixed, assuming that $\lambda_3 > 0$ and ignoring constant terms, (3) can be written as (4), where $\boldsymbol{\mathcal{X}}$ is the matrix of linear predictors,

$$\boldsymbol{\mathcal{X}} = \boldsymbol{X} \sum_{j=1}^{p} (e_j e_j^\top) \beta_j,$$

and P is a sparsity penalty on $\boldsymbol{W}$, given in (5). Furthermore, to assume that there are not overlapping groups (and each variable belongs to exactly one group) can be written as a constraint in $\boldsymbol{W}$, $\|\boldsymbol{W}_{j\cdot}\|_0 = 1$, for every $j = 1, 2 \ldots p$. We will deal with this constraint later, but first let's tackle problem (4).

Problem (4) is a special type of *regularized Singular Value Decomposition*, where the penalty term can be separated into a sum of penalties on the columns of $\boldsymbol{W}$. An efficient way of dealing with this problem, is solving regularized one-rank approximation problems to construct $\boldsymbol{W}$ and $\boldsymbol{T}$ column-wise. An example of such an algorithm is the *sPCA-rSVD* from (Shen and Huang, 2008, Algorithm 1). Our approach here is very similar to theirs, except for the penalty term.

Consider the simpler problem,

$$\min_{\boldsymbol{u},\boldsymbol{v}} \left\{ \left\| \boldsymbol{\mathcal{X}} - \boldsymbol{u}\boldsymbol{v}^\top \right\|_F^2 + \gamma \left(\sum_{j=1}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{v}_j \neq 0) \left\| \boldsymbol{v} \right\|_0 \right)^{1/2} \right\}, \tag{15}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are columns of $\boldsymbol{T}$ and $\boldsymbol{W}$, respectively, as described in Agorithm 1 and 2.

Although the regularization in (15) is discontinuous, an iterative solution is possible, and it is shown in Proposition 3.

Proposition 3 *The optimal* $\boldsymbol{v}$ *in* (15) *is such that, for* $l = 1, 2 \ldots p$,

$$\begin{aligned} \boldsymbol{v}_l & \\ &= (\boldsymbol{\mathcal{X}}^\top \boldsymbol{u})_l \mathbb{1} \Bigg((\boldsymbol{\mathcal{X}}^\top \boldsymbol{u})_l^2 \\ &> \gamma \left(C_{\beta,v}^{(-l)} + \beta_l^2 \right)^{1/2} \\ &\left(C_v^{(-l)} + 1 \right)^{1/2} \\ &- \gamma \left(C_{\beta,v}^{(-l)} C_v^{(-l)} \right)^{1/2} \Bigg), \end{aligned} \tag{16}$$

where

$$C_{\beta,v}^{(-l)} = \sum_{\substack{j=1 \\ j\neq l}}^{p} \beta_j^2 \mathbb{1}(\boldsymbol{v}_j \neq 0), \quad C_v^{(-l)} = \sum_{\substack{j=1 \\ j\neq l}}^{p} \mathbb{1}(\boldsymbol{v}_j \neq 0).$$

The update function for $\boldsymbol{v}$ in Proposition 3 can not be applied in one step, because the expression for each component $\boldsymbol{v}_l$ can not be separated from the whole vector $\boldsymbol{v}$. To tackle this, we propose to iterate through $\boldsymbol{v}$, updating each $\boldsymbol{v}_l$ with (16) until convergence. Algorithm 1 describes the iterative optimization to solve (15), which is a special case of one-rank regularized singular value decomposition. The whole process to find all the columns of $\boldsymbol{W}$ and $\boldsymbol{T}$ is explained in Algorithm 2.

Algorithm 1: One-rank regularized singular value decomposition (*1rSVD*).

```
Result: $\boldsymbol{u}, \boldsymbol{v}$ that minimize (15)
Input: $\boldsymbol{\mathcal{X}}, \boldsymbol{\beta}$
Compute $\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, s$ that minimize $\|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{u}} s \hat{\boldsymbol{v}}^\top\|_F^2$ (one-rank SVD).
Initialize $\boldsymbol{u} \leftarrow \hat{\boldsymbol{u}}$; $\boldsymbol{v} \leftarrow s\hat{\boldsymbol{v}}$
while $\boldsymbol{v}$ not stationary do
    Update $\boldsymbol{v}$ with (16), cyclically iterating component-wise until convergence.
    Update $\boldsymbol{u} \leftarrow \boldsymbol{\mathcal{X}}\boldsymbol{v} / \|\boldsymbol{\mathcal{X}}\boldsymbol{v}\|_2$
end
```

Algorithm 2: Regularized singular value decomposition.

```
Result: $\boldsymbol{W}, \boldsymbol{T}$ that minimize (4)
Input: $\boldsymbol{\mathcal{X}}, \boldsymbol{\beta}, K$
for $k = 1 \dots K$ do
    $\boldsymbol{u}, \boldsymbol{v} \leftarrow$ 1rSVD($\boldsymbol{\mathcal{X}}, \boldsymbol{\beta}$) (Solve the one-rank SVD problem)
    Set $\boldsymbol{T}_k \leftarrow \boldsymbol{u}$; $\boldsymbol{W}_k \leftarrow \boldsymbol{v}$
    $\boldsymbol{\mathcal{X}} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{u}\boldsymbol{v}^\top$ (update $\boldsymbol{\mathcal{X}}$ with the residuals)
end
```

Finally, we have to deal with the non-overlapping groups restriction $\|\boldsymbol{W}_{j\cdot}\|_0 = 1$. We propose a greedy approach to force $\boldsymbol{W}$ to have the desired structure. The idea is to update $\boldsymbol{W}$ by,

$$\boldsymbol{W}_{jk} \leftarrow \boldsymbol{W}_{jk} \mathbb{1}\left(|\boldsymbol{W}_{jk}| = \max_i |\boldsymbol{W}_{ji}|\right), \text{ for all } j, k \tag{17}$$

For sufficiently large values of the penalization hyper-parameter γ, most of the components of $\boldsymbol{W}$ will be zero, so the effect of update (17) will be negligible as γ increases. Update (17) guarantees that $\|\boldsymbol{W}_{j\cdot}\|_0 \leq 1$. To get the equality, we will append $\boldsymbol{W}$ a column $\boldsymbol{W}_{K+1}$ such that $\boldsymbol{W}_{j\,K+1} = \prod_{k=1}^{K} \mathbb{1}(W_{jk} = 0)$, and $\boldsymbol{T}$ a null column $\boldsymbol{T}_{K+1}$.

4 Simulations

This simulation set-up is described in Witten et al. (2014). The data is simulated according to the linear model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with $p = 1000$ features, and ϵ_i i.i.d. from a $N(0, 2.5^2)$ distribution $(1 \leq i \leq n)$. The data matrix $\boldsymbol{X}$ is simulated from a multivariate $N(\mathbf{0}, \mathbf{\Sigma})$ distribution, where $\mathbf{\Sigma} \in \mathbb{R}^{p \times p}$ is block diagonal, given by

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{\boldsymbol{\rho}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{\boldsymbol{\rho}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{1000 \times 1000},$$

with $\mathbf{\Sigma}_{\boldsymbol{\rho}} \in \mathbb{R}^{50 \times 50}$ such that

$$\mathbf{\Sigma}_{\boldsymbol{\rho}}(i,j) = \begin{cases} 1 & i = j \\ \rho & i \neq j \end{cases}.$$

The parameter ρ is varied from 0 to 0.8, exploring different scenarios for the correlation inside groups. The true coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^p$ is random, given by,

$$\boldsymbol{\beta} = [\beta_1\ \beta_2\ \ldots\ \beta_{25}\ \underbrace{0\ \ldots\ 0}_{25}\ \beta_{51}\ \beta_{52}\ \ldots\ \beta_{75}\ \underbrace{0\ \ldots\ 0}_{925}],$$

where

$$\beta_j \sim \begin{cases} U[0.9, 1.1], & 1 \leq j \leq 25 \\ U[-1.1, -0.9], & 51 \leq j \leq 75 \end{cases}.$$

The data matrix is composed of two groups of 50 variables, correlated within each group and independent between groups. Only 25 columns within each group are significant. Additionally, there are another 900 variables that are independent of each other and have no impact on the response. This simulation scheme, as Witten et al. mention, is motivated by gene pathways, where genes within the same pathway have correlated levels of expression, but only a fraction of these are associated with the response of interest.

Table 1 reports the results of the different methods in these simulations. We compared Lasso (Tibshirani, 1996), Ridge, Elastic Net (EN) (Friedman et al., 2010b), Elastic Net Cluster (CEN) (Witten et al., 2014), CEN with known groups, Cluster Group Lasso (Bühlmann et al., 2013), Group Lasso with known groups (Friedman et al., 2010a), and our approach the GLASP. CEN and Group Lasso with known groups have been included for baseline comparisons since groups are, in general, unknown. A training data set composed of 200 observations was used to compare the different algorithms, whereas the hyperparameters were chosen using a validation sample also of size 200. The experiments were repeated 30 times in order to obtain more relevant results, calculated on an independent test sample of 800 observations. The different algorithms have been compared in terms of root mean squared error (RMSE) of the linear predictor, i.e,

$$\text{RMSE} = \left\| \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}\hat{\boldsymbol{\beta}} \right\|_2 .$$

We have also studied the accuracy of the variable selection (Correct Zeros), the number of coefficients different from zero (notice that 50 is the correct number of coefficients different from zero in the generating model), as well as the Rand Index (RI) (Rand, 1971), which measures the agreement between the actual and estimated clusters with each algorithm. This index varies between 0 and 1 (from low to high agreement). In the case of Lasso, Ridge, and EN, the reported groups are found by the k-means algorithm applied to the linear predictor matrix after estimating $\hat{\boldsymbol{\beta}}$. The values reported in Table 1 correspond to the means in 30 repetitions. The standard errors of the mean are shown in parentheses.

The results in Table 1 show that GLASP is superior to the other methods (except for the baseline methods with known groups) in terms of RMSE and Correct Zeros, sometimes by a large margin, when correlations within groups are moderate (0.1, 0.2, 0.5). Furthermore, in general, the number of non-zero coefficients selected by GLASP is the lowest among the different methods, resulting in more parsimonious models. Concerning the Rand Index, GLASP is usually lower than other approaches, but we believe this is because GLASP builds the groups by balancing both criteria, the correlations between predictors and the relationship between predictors and the response variable. Therefore, GLASP does not produce either of the two groupings that are trivial in this case: two groups of 50 and one group of 900 (correlation), or two groups of 25 and one group of 950 (prediction). Groups found by GLASP are closely related to those groups that one can compute from the singular value decomposition (or, equivalently, the principal components) of the matrix of linear predictors.

Table 1: Average results of GLASP and other methods on a test set (800 observations) over 30 simulations. Standard errors are given in parenthesis. Models were fit on a training set (200 observations) with the hyperparameters that led to optimal RMSE on a validation set (200 observations). CEN and Group Lasso with known groups have been included for baseline comparisons (shaded rows).

		$\boldsymbol{\rho = 0.0}$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	166.662(1.532)	0.885(0.005)	133.967(6.35)	0.909(0.001)
Ridge + Kmeans	182.004(0.97)	0.05(0)	1000(0)	0.897(0.004)
EN + Kmeans	165.491(1.264)	0.831(0.013)	196.767(13.684)	0.909(0.001)
CEN	167.013(1.463)	0.777(0.032)	253.933(32.999)	0.908(0)
CEN Known Groups	162.681(1.282)	0.807(0.008)	224.767(9.708)	1(0)
Cluster Group Lasso	183.714(0.871)	0.05(0)	1000(0)	0.366(0)
Group Lasso Known Groups	56.759(1.277)	0.113(0.044)	936.667(44.005)	1(0)
GLASP	172.771(1.414)	0.67(0.047)	358.633(49.427)	0.774(0.013)
		$\boldsymbol{\rho = 0.1}$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	93.876(2.304)	0.911(0.004)	138.533(4.245)	0.951(0.003)
Ridge + Kmeans	199.147(1.619)	0.05(0)	1000(0)	0.949(0.002)
EN + Kmeans	93.635(2.283)	0.906(0.004)	143.5(4.127)	0.953(0.003)
CEN	93.954(2.357)	0.892(0.011)	157.433(11.147)	0.953(0.003)
CEN Known Groups	91.809(2.117)	0.881(0.011)	169.167(10.995)	1(0)
Cluster Group Lasso	166.879(2.137)	0.154(0.032)	895.433(32.323)	0.395(0.004)
Group Lasso Known Groups	39.468(0.866)	0.335(0.081)	715(80.841)	1(0)
GLASP	90.545(2.669)	0.956(0.009)	87(9.184)	0.914(0.003)
		$\boldsymbol{\rho = 0.2}$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	77.449(1.807)	0.933(0.004)	116.4(3.753)	0.98(0.001)

Ridge + Kmeans	185.387(1.779)	0.05(0)	1000(0)	0.936(0.001)
EN + Kmeans	77.166(1.768)	0.931(0.004)	118.467(3.907)	0.981(0.001)
CEN	73.654(1.306)	0.744(0.026)	306.267(25.934)	0.983(0.002)
CEN Known Groups	74.051(1.5)	0.829(0.017)	221.4(16.967)	1(0)
Cluster Group Lasso	77.141(3.553)	0.104(0.037)	946.5(37.447)	0.839(0.021)
Group Lasso Known Groups	35.551(0.775)	0.43(0.086)	620(86.423)	1(0)
GLASP	62.03(1.517)	0.97(0.005)	79.333(5.137)	0.943(0.002)
		$\rho = \mathbf{0.5}$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	64.632(1.583)	0.958(0.002)	91.933(2.505)	0.982(0.001)
Ridge + Kmeans	149.217(1.627)	0.05(0)	1000(0)	0.91(0)
EN + Kmeans	63.369(1.407)	0.946(0.003)	103.7(3.077)	0.984(0.001)
CEN	61.36(1.52)	0.789(0.049)	260.567(49.496)	0.988(0.001)
CEN Known Groups	52.998(1.119)	0.813(0.022)	236.667(22.255)	1(0)
Cluster Group Lasso	59.377(0.888)	0.2(0.062)	850(62.284)	0.906(0)
Group Lasso Known Groups	29.144(0.691)	0.905(0.053)	145(52.923)	1(0)
GLASP	58.516(1.757)	0.968(0.002)	82.3(1.675)	0.963(0.003)
		$\rho = \mathbf{0.8}$		
Method	RMSE	Correct Zeros	Num. Non-Zeros	RI
Lasso + Kmeans	60.031(1.348)	0.955(0.002)	89.7(2.476)	0.964(0.002)
Ridge + Kmeans	114.86(1.454)	0.05(0)	1000(0)	0.906(0)
EN + Kmeans	52.864(0.94)	0.935(0.003)	114.033(2.943)	0.969(0.001)
CEN	42.833(0.879)	0.732(0.043)	317.9(43.026)	0.993(0.001)
CEN Known Groups	32.346(0.834)	0.753(0.048)	296.867(48.34)	1(0)
Cluster Group Lasso	48.994(0.493)	0.17(0.057)	880(56.812)	0.906(0)
Group Lasso Known Groups	20.968(0.692)	0.905(0.053)	145(52.923)	1(0)
GLASP	48.291(0.892)	0.954(0.001)	96.333(0.946)	0.987(0.002)

5 Extension to other models

In Section 4, the choice of the function $L(\boldsymbol{\beta})$ corresponds to classical linear models. However, one strength of our methodology is that it can easily extend other risk functions, such as logistic regression or Cox models.

We have implemented the following three types of problems: linear and logistic regression and Cox proportional hazard models with right-censoring. In the first two cases, the function L is given by,

- *Linear regression*

$$L(\boldsymbol{\beta}) = \frac{1}{N} \left\| \boldsymbol{y} - \boldsymbol{\eta} \right\|_2^2 ,$$

  where $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$ is the linear predictor.
- *Logistic regression*

$$L(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{\eta_i}\right) - \boldsymbol{y}_i \eta_i .$$

Our implementation requires to determine the gradient of L, ∇L, which is given in each case by,

- *Linear regression*

$$\nabla L(\boldsymbol{\beta}) = -\frac{1}{N} \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\eta}).$$

– *Logistic regression*

$$\nabla L(\boldsymbol{\beta}) = \frac{1}{N}\boldsymbol{X}^\top \left(\frac{1}{1+e^{\boldsymbol{\eta}}} - \boldsymbol{y}\right)$$

There are lots of numerical details to consider, especially in the case of logistic regression. For example, the function $\log(1 + e^{\eta})$ is unstable when $|\eta|> 30$. However, it can be substituted by a more stable approximation, given by

$$\hat{\log}(1+e^{\eta}) = \begin{cases} \eta, & \eta > 33.3 \\ \eta + e^{-\eta}, & 18 < \eta < 33.3 \\ \log(1+e^{\eta}), & -37 < \eta < 18 \\ e^{\eta}, & \eta < -37 \end{cases} \tag{18}$$

Similarly, its derivative can be replaced by

$$\frac{d}{d\eta}\hat{\log}(1+e^{\eta}) = \begin{cases} (1+e^{-\eta}, & \eta > -30 \\ e^{\eta}, & \eta < -30 \end{cases} \tag{19}$$

Although our implementation can address logistic regression, we will now focus on the Cox model, which has been less addressed in the literature from the perspective of variable selection.

5.1 Proportional hazards model with right-censoring

Under the proportional hazards model framework with right-censoring, we assume we have a covariate matrix $\boldsymbol{X} \in \mathbb{R}^{N\times p}$, a vector of event times $\boldsymbol{t} \in \mathbb{R}^{N\times 1}$ and a vector of event indicator $\boldsymbol{\delta} \in \mathbb{R}^{N\times 1}$ ($\delta_i = 1$ if an event was observed at time t_i, and $\delta_i = 0$ if time t_i is right-censored).

The proportional hazards model assumption states that, for an individual with covariates $\boldsymbol{x}^\top \in \mathbb{R}^{1\times p}$, their hazard function $h(t)$ is given by

$$h(t) = h_0(t)\exp(\boldsymbol{x}^\top\boldsymbol{\beta}),$$

where $h_0(t)$ is a baseline hazard function. This is a semi-parametric model, because $h_0(t)$ is not assumed to have a particular parametric form. More details can be found in Moore (2016).

In the case of right censoring, our function L is the negative log-partial likelihood and it is given by,

$$L(\boldsymbol{\beta}) = \sum_{i\in D}\boldsymbol{x}_i^\top\boldsymbol{\beta} - \sum_{i\in D}\log\left(\sum_{k\in R_i}\exp(\boldsymbol{x}_k^\top\boldsymbol{\beta})\right),$$

where D is the index set of observed events, and R_i is the index set of individuals at risk at time t_i. Furthermore, the first derivative of L has the expression,

$$\frac{\partial}{\partial\beta_j}L(\boldsymbol{\beta}) = \sum_{i\in D}\left(\boldsymbol{x}_{ij} - \frac{\sum_{k\in R_i}\boldsymbol{x}_{kj}\exp(\boldsymbol{x}_k^\top\boldsymbol{\beta})}{\sum_{k\in R_i}\exp(\boldsymbol{x}_k^\top\boldsymbol{\beta})}\right).$$

Once the model is fitted, with coefficient vector $\hat{\boldsymbol{\beta}}$, to estimate the baseline survival function we use,

$$S_0(t) = \exp(-H_0(t)), \text{ with } H_0(t) = \sum_{t_i \leq t} h_0(t_i),$$

where

$$h_0(t_i) = \frac{\delta_i}{\sum_{j \in R_i} \exp(\boldsymbol{x}_j^\top \hat{\boldsymbol{\beta}})}.$$

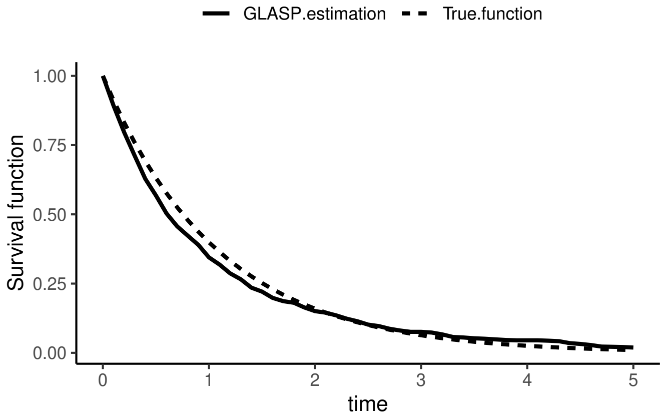
An individual's estimated survival function is given by

$$S(t|\boldsymbol{x}) = S_0(t)^{\exp(\boldsymbol{x}^\top \hat{\boldsymbol{\beta}})}. \tag{20}$$

Example 1 *For illustrative purposes, we simulated a survival data set. The data matrix* $\boldsymbol{X} \in \mathbb{R}^{1000\times 10}$ *has i.i.d.* $N(0,1)$ *columns and* $\boldsymbol{\beta}_j \sim N(0,1/9)$ *for* $j \leq 5$ *and* 0 *otherwise. The underlying survival time* t_i^* *for a row* $\boldsymbol{x}_i^\top$ *is simulated exponential with parameter* $\lambda = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})$*. The censoring time* s_i *distributes exponential with parameter* $\lambda = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})/2$*. The observed time is the minimum between* t_i^* *and* s_i*.*

Figure 2 displays the estimation of the survival function given by (20)*, for an individual outside the training sample. In this case, the estimated function is remarkably close to the true survival function of this individual, according to the simulated model.*

Fig. 2 Survival functions $S(t|\boldsymbol{x})$ estimated and real for an individual with simulated covariates. The GLASP model has been fitted on simulated data with the same $\boldsymbol{x}$ distribution.



5.2 Simulation studies: right-censored survival data

We consider an adaptation of the simulation set-ups described in Section 4. This time the response variable $\boldsymbol{t}$ and the event indicator $\boldsymbol{\delta}$ are simulated, for every $i = 1, 2 \dots N$, according to the scheme,

$$\begin{aligned} h_i &= \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}), \\ t_i^* &\sim Exp(\lambda = h_i), \\ s_i &\sim Exp(\lambda = h_i/2), \\ t_i &= \min(t_i^*, s_i), \\ \delta_i &= \mathbb{1}(t_i = t_i^*). \end{aligned}$$

The data matrix $\boldsymbol{X}$ is simulated from a multivariate $N(\mathbf{0}, \boldsymbol{\Sigma})$ distribution, where $\boldsymbol{\Sigma}$ is block diagonal, given by

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma_\rho} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma_\rho} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{20\times 20},$$

with $\boldsymbol{\Sigma_\rho} \in \mathbb{R}^{5\times 5}$ such that

$$\boldsymbol{\Sigma_\rho}(i,j) = \begin{cases} 1 & i = j \\ 0.5 & i \neq j \end{cases}.$$

The true coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^p$ is random, given by,

$$\boldsymbol{\beta} = [\beta_1 \; \beta_2 \; 0 \; 0 \; 0 \;\; \beta_6 \; \beta_7 \; \underbrace{0 \; \dots \; 0}_{13}],$$

where

$$\beta_j \sim \begin{cases} U[0.9, 1.1], & 1 \leq j \leq 2 \\ U[-1.1, -0.9], & 6 \leq j \leq 7 \end{cases}.$$

In this case, the X matrix has two significant groups of 5 variables, and only 2 variables within each group have an actual impact on the generating model. We have simulated 50 observations for training and 50 for testing. Furthermore, to obtain relevant results, the simulations were repeated 30 times, and the results averaged.

The models studied in Section 4 no longer apply, as they are not studied for Cox regression, or do not have an effective method for the selection of the regularization hyperparameters in the case of survival data. We have compared GLASP and the function `coxph` of the R package `survival` (Therneau, 2015; Therneau and Grambsch, 2000). To calculate the groups given by `coxph`, we have used k-means, applied to the matrix of linear predictors once the model was adjusted, as we did in Section 4 for the algorithms that would not directly compute variable clusters.

Table 2 highlights the results of the simulations for survival data. The metric $\boldsymbol{\beta}$ WMSE (weighted mean squared error) refers to the $\boldsymbol{\beta}$ estimation error, given by $(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \Sigma (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$, as described in Zhao et al. (2019). The rates $\boldsymbol{\beta}$ TPR (true positive rate) and TNR (true negative rate) refer to the correct identification of the variables that enter the model. Moreover, we included in Table 2 the mean estimation error of the survival curve $S(t|\boldsymbol{x})$ for the individuals in the test sample, measured as the integral of the absolute difference of the estimated and actual curves for each individual.

One can see from Table 2 that the estimation of GLASP is superior to the classical estimation of `coxph` in almost every aspect. We believe that this difference is, apart from the algorithm itself, also accentuated by the regularization hyperparameter selection approach that we have integrated with GLASP, described in the next section.

Table 2: Average results of GLASP and the Cox proportional hazards model.

Method	$\boldsymbol{\beta}$ WMSE	Correct Zeros	$\boldsymbol{\beta}$ TPR	$\boldsymbol{\beta}$ TNR	Num. Non-Zeros	$S(t\|x)$ error	RI
coxph	9.11 (1.54)	0.2 (0)	1 (0)	0 (0)	20 (0)	15.88 (0.35)	0.53 (0.01)
GLASP	3.27 (0.61)	0.37 (0.04)	0.95 (0.03)	0.23 (0.06)	16.03 (1.01)	13.86 (0.43)	0.58 (0.03)

5.3 Simulation studies: logistic regression models

In this section, we consider another adaptation of the simulation set-ups described in Section 4. This time, the response variable $\boldsymbol{y}$ is simulated, for every $i = 1, 2 \dots N$ according to the Bernoulli model,

$$Y_i \sim Ber((1 + \exp(-\eta))^{-1}),$$

where $\eta = \boldsymbol{\beta}^\top \boldsymbol{x}_i$. The simulation set-up parameters $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}_\rho$ have been chosen as in Section 4. A total of 200 observations have been used as training, and 650 as testing. For baseline comparison, we have included elastic-net (glmnet) and lasso. The hyperparameters of GLASP, glmnet and lasso were selected using four fold cross-validation and random search on the training sample. More details about the implementation and how to optimally select hyperparameters of GLASP, glmnet and lasso using a unified framework can be found in Section 6. Table 3 displays the average test accuracy (ACC) and area under the ROC curve (AUC) of the models estimated with GLASP, glmnet and lasso. To study the convergence of the optimization algorithm of GLASP in the logistic case, we have measured the number of iterations (Num. Iter.) of the outer loop of the algorithm (that alternatively solves (2) and (3) until convergence). As shown in Table 3, when $\rho = 0$ the method converges slower on average (6 iterations versus 3), which is aligned with what we expect when there is no clear group structure between the variables. On the other hand, Table 3 confirms that GLASP outperforms glmnet and lasso in terms of ACC, AUC and correct selection of variables, when the correlation inside groups is moderate. These results are aligned with those in Table 1, when the response is linear. When there is no correlation ($\rho = 0$) the theoretical assumptions of GLASP are not met, but even in that worse case scenario GLASP is a competitive alternative to lasso and glmnet.

Table 3: Average results of Logistic GLASP, glmnet and lasso on a test set (650 observations) over 30 simulations. Standard errors are given in parenthesis and significantly superior results (two standard errors) are bold. Models were fit on a training set (200 observations) using four-fold cross validation and random search for the hyperparameter optimization, and the model selection was based on the validation accuracy.

			$\rho = 0$			
Method	ACC	AUC	Correct Zeros	Num. Non-Zeros	RI	Num. Iter.
lasso	0.592(0.01)	0.631(0.01)	**0.863**(0)	**139**(1.2)	-	-
glmnet	**0.618**(0.01)	**0.668**(0.01)	0.679(0.02)	345(21)	-	-
GLASP	0.605(0.01)	0.647(0.01)	0.785(0.04)	204(49)	0.057(0.01)	6.23(0.9)
			$\rho = 0.1$			
Method	ACC	AUC	Correct Zeros	Num. Non-Zeros	RI	Num. Iter.
lasso	0.783(0)	0.870(0)	**0.9**(0)	**111**(1)	-	-
glmnet	**0.816**(0)	**0.901**(0)	0.724(0.01)	315(15)	-	-
GLASP	**0.817**(0.01)	**0.901**(0.01)	**0.901**(0.02)	**77.5**(27)	0.237(0.05)	2.83(0.3)
			$\rho = 0.2$			
Method	ACC	AUC	Correct Zeros	Num. Non-Zeros	RI	Num. Iter.
lasso	0.835(0)	0.923(0)	0.911(0)	97.9(1)	-	-
glmnet	0.868(0)	0.948(0)	0.739(0.01)	301(14)	-	-
GLASP	**0.888**(0)	**0.962**(0)	**0.928**(0.02)	**42.7**(21)	0.386(0.07)	2.97(0.1)
			$\rho = 0.5$			
Method	ACC	AUC	Correct Zeros	Num. Non-Zeros	RI	Num. Iter.
lasso	0.898(0)	0.970(0)	0.919(0)	**74.8**(1)	-	-
glmnet	0.925(0)	0.984(0)	0.786(0.01)	252(13)	-	-
GLASP	**0.931**(0)	**0.987**(0)	**0.939**(0.01)	**60.9**(11)	0.344(0.07)	3.4(0.2)
			$\rho = 0.8$			
Method	ACC	AUC	Correct Zeros	Num. Non-Zeros	RI	Num. Iter.
lasso	0.930(0)	0.986(0)	**0.915**(0)	**63.8**(1)	-	-
glmnet	**0.949**(0)	**0.993**(0)	0.820(0.01)	215(13)	-	-
GLASP	**0.948**(0)	**0.993**(0)	0.899(0.02)	118(25)	0.247(0.06)	3.3(0.3)

6 Implementation details

In this section, we will describe some details of the implementation of GLASP, with emphasis on its R interface, and the selection of hyperparameters.

6.1 Interface

Recently, Kuhn and Vaughan have developed the R `parsnip` package (Kuhn and Vaughan, 2020), which provides a standard and organized interface for creating modelling packages in R. A critical advantage of this approach is that it integrates very well with other `tidymodels` packages. We have implemented our GLASP algorithm in an R package called `glasp`[1], created with the vision to integrate with `parsnip`, as well as the rest of `tidymodels` packages. This offers numerous advantages, highlighting, for example, the optimization of hyperparameters, which is always a concern in penalized models.

[1] `https://github.com/jlaria/glasp`

All the internal optimization of the glasp library has been implemented in C++, and integrated in R through RcppArmadillo (Eddelbuettel and François, 2011) and Rcpp (Eddelbuettel and François, 2011).

Currently, the parsnip library considers two types of objectives for the models: regression and classification. Taking this into account, we have created parsnip models for each task, namely glasp_regression, glasp_classification, glasp_cox. For example, one way to fit a GLASP model for linear regression would be as follows.

```
glasp_regression() %>% set_engine("glasp") %>% fit(y~., data)
```

To adjust logistic regression is analogous, changing glasp_regression() to glasp_classification(), and glasp_cox to specify a survival model. In the case of Cox regression, the response variable is $\boldsymbol{\delta}$ (indicating whether the event has been observed at each instant of time), and the covariates are the columns of $\boldsymbol{X}$ and the instants of time $\boldsymbol{t}$. Thus, a GLASP model for Cox regression with right-censoring would fit as follows.

```
glasp_cox() %>% set_engine("glasp") %>% fit(event ~ time + ., data)
```

Let $\hat{\boldsymbol{\beta}}$ be the coefficient estimation obtained by GLASP. Since (20) provides an approximation of the survival function, then the probability of having observed the event at a time prior to t for an individual with covariates $\boldsymbol{x}$, can be estimated as $p(t, \boldsymbol{x}) = 1 - S(t|\boldsymbol{x})$. One would expect $p(t_i, \boldsymbol{x}_i)$ to be high if $\delta_i = 1$ and low if $\delta_i = 0$, therefore, in a very practical predictive context, a survival problem can be considered as a classification problem, where $p(T, X) \approx P(\delta = 1)$. An important advantage of considering it this way, is that all predictive error metrics associated to classification problems (accuracy, sensitivity, specificity, F1-score, etc) can be calculated in survival problems, which provides a general approach to optimize hyperparameters.

6.2 Hyperparameter selection

Since glasp_model is a parsnip model, it integrates with the tune and dials libraries (Kuhn, 2020) to deal with the optimization of the hyperparameters λ_1, λ_2, λ_3, and K, from a very general approach. Package tune offers implementations of the three most popular types of hyperparameter search: grid search, random search (Bergstra and Bengio, 2012) and Bayesian Optimization (Snoek et al., 2012).

For example, the following code in R finds the optimal combination of hyperparameters that minimizes the area under the ROC curve for a GLASP model in simulated survival data, using Bayesian Optimization, and 4-fold cross validation.

```
data <- simulate_dummy_surv_data()
model <- glasp_cox(l1 = tune(),
                   l2 = tune(),
                   frob = tune(),
                    num_comp = tune()) %>%
          set_engine("glasp")
data_rs <- vfold_cv(data, v = 4)
hist <- tune_bayes(model, event~.,
                   resamples = data_rs,
                   metrics = metric_set(roc_auc),
                   iter = 100)
show_best(hist, metric = "roc_auc")
```

In the simulation studies of Sections 4 and 5, the GLASP hyperparameters were optimized using random search.

7 Application to right-censored survival data

In this section we present an application of GLASP to real data from a study of patients with diffuse large-B-cell lymphoma (DLBCL). The data is available as right-censored survival sample data in the BioNet packages. For more information see Dittrich et al. (2008), Beisser et al. (2010), and Alizadeh et al. (2000).

The study of gene-expression profiles as predictors for survival of patients with DLBCL is motivated by the large variation in survival times after treatment of this disease, even for patients with similar clinical features. Several authors have studied patients with DLBCL, trying to predict the survival of individuals receiving treatment based on high-dimensional microarray gene expression data. Among these, we can find the works of Rosenwald et al. (2002), Bair et al. (2006) and Chen et al. (2011). To pre-process the data, we have followed an approach similar to that of Chen et al. (2011). We selected the genes for which individual Cox scores, obtained after fitting univariate Cox regression models, were more significant than a certain threshold. After removing missing values, the data were composed of 190 observations, 78 genetic features, and one clinical variable, which is a factor variable with several levels.

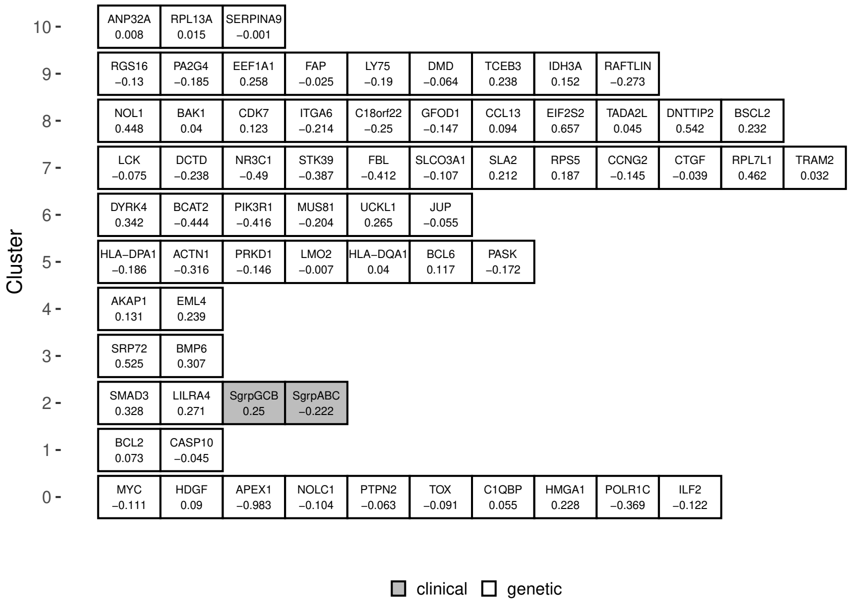
The objective of using the GLASP methodology with this dataset is twofold. Firstly, to build a survival model that includes only relevant genetic and clinical characteristics. Secondly, to find clusters among those relevant features, as GLASP can reveal hidden biological interrelations between gene expressions associated with this particular disease.

Figure 3 depicts the resulting coefficient estimation and feature clustering from GLASP in the DLBCL survival data described above. The output in Figure 3 includes only those variables with associated non-zero coefficients. According to the model, there are 11 groups, with varying sizes. From a biological perspective, the resulting clustering could give insight into possible genetic interactions. For example, Cluster 1 includes BCL2 and CASP10. Both genes are associated with cell apoptosis. BCL2 blocks the apoptotic death of some cells such as lymphocytes[2], whereas CASP10 plays a central role in the

[2] `https://www.genecards.org/cgi-bin/carddisp.pl?gene=BCL2`

execution-phase of cell apoptosis[3]. This not only explains that they are in the same group, but also that their associated coefficients have opposite sign. As another illustration, Cluster 3, is formed by BMP6 and SRP72. BMP6 induces cartilage and bone formation[4], and mutations of SRP72 are associated with familial bone marrow failure[5].

Fig. 3 GLASP model estimation and gene-clustering for the diffuse large-B-cell lymphoma dataset. Each row represents a cluster, with squares describing each variable that was included in the final model and its associated coefficient estimation in the Cox model.



8 Conclusions

The main contribution of this paper is the formulation of GLASP, a supervised variable clustering method, very competitive not only as a clustering method but also as a predictive model. Multiple models have been unified under a joint implementation, which also integrates with the latest algorithms for hyperparameter search in R. The methodologies are rarely so flexible that they allow adjusting classification problems, regression, and Cox survival models with the same algorithm. Moreover, Section 7 showcased an application of GLASP to

[3] https://www.genecards.org/cgi-bin/carddisp.pl?gene=CASP10

[4] https://www.genecards.org/cgi-bin/carddisp.pl?gene=BMP6&keywords=BMP6

[5] https://www.genecards.org/cgi-bin/carddisp.pl?gene=SRP72&keywords=SRP72

biological survival data. From a methodological point of view, this paper has also introduced a particular case of sparse Singular Value Decomposition, with a penalty term appearing naturally from the Group Lasso penalty. Its solution and implementation using coordinate-descend was demonstrated in detail.

In the simulation studies in sections 4 and 5, it is observed that GLASP is substantially advantageous in terms of predictive ability and variable selection, apart from providing the simplest models. In the simulations of Section 4 we noticed that GLASP is the preferred alternative when the correlations between variables of the same group are moderate. If the dependencies are low, all the methods have similar performance, whereas if the correlations are high, Cluster Elastic Net has better performance.

Regarding possible extensions, we propose to explore possible safe-rules that would rule out multiple predictors from the very beginning, in order to reduce the dimension of the problem, as Ndiaye et al. (2016); Tibshirani et al. (2012) do. Moreover, this would also allow finding bounds for the regularization hyperparameters, and thus accelerate the search for the best combinations.

Acknowledgements We gratefully acknowledge the help provided by Prof. Daniela Witten, who gave us access to the source code of CEN and the simulation set-ups compared in Section 4. We also acknowledge the constructive comments of the anonymous referees that have contributed to improve the contents of this paper.

Supplementary Material

Appendix: Proof of Propositions 2, 3 and 1. (PDF)

R-package glasp: R-package glasp containing code of the method described in the article. (GNU zipped tar file)

Source code: R scripts to generate Figures 1, 2, and 3, as well as Tables 1 and 2. The data set studied in Section 7 is included. These files are also available at `https://github.com/jlaria/glasp-code`. The code is shipped with a docker-compose file to replicate the exact R environment used in this paper with an rstudio web server interface. (Zip archive)

References

Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A, Boldrick JC, Sabet H, Tran T, Yu X, et al. (2000) Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature 403(6769):503–511

Bair E, Hastie T, Paul D, Tibshirani R (2006) Prediction by supervised principal components. Journal of the American Statistical Association 101(473):119–137

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences 2(1):183–202

Beisser D, Klau GW, Dandekar T, Müller T, Dittrich MT (2010) Bionet: an r-package for the functional analysis of biological networks. Bioinformatics 26(8):1129–1130

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. Journal of Machine Learning Research 13(Feb):281–305

Bühlmann P, Rütimann P, van de Geer S, Zhang CH (2013) Correlated variables in regression: clustering and sparse estimation. Journal of Statistical Planning and Inference 143(11):1835–1858

Chen K, Chen K, Müller HG, Wang JL (2011) Stringing high-dimensional data for functional analysis. Journal of the American Statistical Association 106(493):275–284

Ciuperca G (2020) Adaptive elastic-net selection in a quantile model with diverging number of variable groups. Statistics 54(5):1147–1170

Dittrich MT, Klau GW, Rosenwald A, Dandekar T, Müller T (2008) Identifying functional modules in protein–protein interaction networks: an integrated exact approach. Bioinformatics 24(13):i223–i231

Eddelbuettel D, François R (2011) Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40(8):1–18, DOI 10.18637/jss.v040.i08, URL `http://www.jstatsoft.org/v40/i08/`

Friedman J, Hastie T, Tibshirani R (2010a) A note on the group lasso and a sparse group lasso. arXiv preprint arXiv:10010736

Friedman J, Hastie T, Tibshirani R (2010b) Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1):1

Kuhn M (2020) tune: Tidy Tuning Tools. URL `https://CRAN.R-project.org/package=tune`, r package version 0.1.0

Kuhn M, Vaughan D (2020) parsnip: A Common API to Modeling and Analysis Functions. URL `https://CRAN.R-project.org/package=parsnip`, r package version 0.0.5

Laria JC, Carmen Aguilera-Morillo M, Lillo RE (2019) An iterative sparse-group lasso. Journal of Computational and Graphical Statistics pp 1–10

Luo S, Chen Z (2020) Feature selection by canonical correlation search in high-dimensional multiresponse models with complex group structures. Journal of the American Statistical Association 115(531):1227–1235

Moore DF (2016) Applied survival analysis using R. Springer

Ndiaye E, Fercoq O, Gramfort A, Salmon J (2016) Gap safe screening rules for sparse-group lasso. In: Advances in Neural Information Processing Systems, pp 388–396

Price BS, Sherwood B (2017) A cluster elastic net for multivariate regression. Journal of Machine Learning Research 18(1):8685–8723

Rand WM (1971) Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336):846–850

Ren S, Kang EL, Lu JL (2020) Mcen: a method of simultaneous variable selection and clustering for high-dimensional multinomial regression. Statistics and Computing 30(2):291–304

Rosenwald A, Wright G, Chan WC, Connors JM, Campo E, Fisher RI, Gascoyne RD, Muller-Hermelink HK, Smeland EB, Giltnane JM, et al. (2002) The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. New England Journal of Medicine 346(25):1937–1947

Shen H, Huang JZ (2008) Sparse principal component analysis via regularized low rank matrix approximation. Journal of Multivariate Analysis 99(6):1015–1034

Simon N, Friedman J, Hastie T, Tibshirani R (2013) A sparse-group lasso. Journal of Computational and Graphical Statistics 22(2):231–245

Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp 2951–2959

Therneau TM (2015) A Package for Survival Analysis in S. URL `https://CRAN.R-project.org/package=survival`, version 2.38

Therneau TM, Grambsch PM (2000) Modeling Survival Data: Extending the Cox Model. Springer, New York

Tibshirani R (1996) Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) 58(1):267–288

Tibshirani R, Bien J, Friedman J, Hastie T, Simon N, Taylor J, Tibshirani RJ (2012) Strong rules for discarding predictors in lasso-type problems. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 74(2):245–266

Witten DM, Shojaie A, Zhang F (2014) The cluster elastic net for high-dimensional regression with unknown variable grouping. Technometrics 56(1):112–122

Zhang Y, Zhang N, Sun D, Toh KC (2020) An efficient hessian based algorithm for solving large-scale sparse group lasso problems. Mathematical Programming 179(1):223–263

Zhao H, Wu Q, Li G, Sun J (2019) Simultaneous estimation and variable selection for interval-censored data with broken adaptive ridge regression. Journal of the American Statistical Association pp 1–13

Zhou N, Zhu J (2010) Group variable selection via a hierarchical lasso and its oracle property. Statistics and Its Interface 3:557–574

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2):301–320