Bioinformatics, 37(5), 2021, 659–668
doi: 10.1093/bioinformatics/btaa855
Advance Access Publication Date: 5 October 2020
Original Paper

OXFORD

Gene expression

selectBoost: a general algorithm to enhance the performance of variable selection methods

Frédéric Bertrand [1,2,*], Ismaïl Aouadi[3,4,5,†], Nicolas Jung[1,3,†], Raphael Carapito[3,4,5], Laurent Vallat[3,5,‡], Seiamak Bahram[3,4,5] and Myriam Maumy-Bertrand[1]

[1]Institut de Recherche Mathématique Avancée, CNRS UMR 7501, Labex IRMIA, Université de Strasbourg, Strasbourg, France, [2]Université de Technologie de Troyes, ICD, ROSAS, M2S, Troyes, France, [3]ImmunoRhumatologie Moléculaire, INSERM UMR_S 1109, LabEx TRANSPLANTEX, Centre de Recherche d'Immunologie et d'Hématologie, Fédération de Médecine Translationnelle de Strasbourg (FMTS), Université de Strasbourg, Strasbourg, France, [4]Laboratoire International Associé (LIA) INSERM, Strasbourg (France) - Nagano (Japan), Strasbourg, France and [5]Fédération Hospitalo-Universitaire (FHU) OMICARE, Laboratoire Central d'Immunologie, Pôle de Biologie, Nouvel Hôpital Civil, Hôpitaux Universitaires de Strasbourg, Strasbourg, France

*To whom correspondence should be addressed.
†The authors wish it to be known that these authors contributed equally.
‡Present address: IRFAC, INSERM UMR_S1113, FMTS, Université de Strasbourg, Strasbourg, France

Associate Editor: Cowen Lenore

Received on July 19, 2019; revised on September 2, 2020; editorial decision on September 19, 2020; accepted on September 21, 2020

Abstract

Motivation: With the growth of big data, variable selection has become one of the critical challenges in statistics. Although many methods have been proposed in the literature, their performance in terms of recall (sensitivity) and precision (predictive positive value) is limited in a context where the number of variables by far exceeds the number of observations or in a highly correlated setting.

Results: In this article, we propose a general algorithm, which improves the precision of any existing variable selection method. This algorithm is based on highly intensive simulations and takes into account the correlation structure of the data. Our algorithm can either produce a confidence index for variable selection or be used in an experimental design planning perspective. We demonstrate the performance of our algorithm on both simulated and real data. We then apply it in two different ways to improve biological network reverse-engineering.

Availability and implementation: Code is available as the `SelectBoost` package on the CRAN, https://cran.r-project.org/package=SelectBoost. Some network reverse-engineering functionalities are available in the Patterns CRAN package, https://cran.r-project.org/package=Patterns.

Contact: frederic.bertrand@utt.fr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Technological innovations make it possible to measure large amounts of data in a single observation. As a consequence, problems in which the number P of variables is larger than the number N of observations have become common. As reviewed by Fan and Li (2006), such situations arise in many fields from fundamental sciences to social science, and variable selection is required to tackle these issues. For example, in biology/medicine, thousands of messenger RNA (mRNA) expressions (Lipshutz *et al.*, 1999) may be potential predictors of some disease. Moreover, in such studies, the correlation between variables is often very strong (Segal *et al.*, 2003), and variable selection methods often fail to make the distinction between the informative variables and those which are not. Similarly, inference of gene regulatory networks from perturbation data can enhance the insights of a biological system (Morgan *et al.*, 2019). In this article, we propose a general algorithm that enhances model selection in correlated variables.

First, we will assume a statistical model with a response variable $\mathbf{y} = (y_1, \ldots, y_N)'$ (with the symbol '' as the transposed), a variable matrix of size $N \times P$, $\mathbf{X} = (\mathbf{x}_{1.}, \ldots, \mathbf{x}_{P.})$ and a vector of parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$. Then, we will assume that the vector of parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$ is sparse. In other words, we will assume that $\beta_i = 0$ except for a quite small proportion of elements of the vector. We note $\mathcal{S}$ as the set of indices for which $\beta_i \neq 0$ and $q < \infty$ is the

© The Author(s) 2020. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

cardinality of this set $\mathcal{S}$. Without any loss of generality, we will assume that $\beta_p \neq 0$ if and only if $p \leq q$.

When dealing with a problem of variable selection, one of the goals is the estimation of the support, in which you want $\mathbb{P}(\mathcal{S} = \hat{\mathcal{S}})$ to be close to one, with $\hat{\mathcal{S}} = \{k : \beta_k \neq 0\}$. Here, our interest is mainly as follows, i.e. in identifying the correct support $\mathcal{S}$. This kind of issue arises in many fields, e.g. in biology, where it is of greatest interest to discover which specific molecules are involved in a disease (Fan and Li, 2006).

There is a vast literature dealing with the problem of variable selection in both statistical and machine-learning areas (Fan and Li, 2006; Fan and Lv, 2010). The main variable selection methods can be gathered in the common framework of penalized likelihood. The estimate $\hat{\boldsymbol{\beta}}$ is then given by:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}}[-\ell_N(\boldsymbol{\beta}) + \sum_{p=1}^{P} \operatorname{pen}_\lambda(\beta_p)], \tag{1}$$

where $\ell_N(.)$ is the log-likelihood function, $\operatorname{pen}_\lambda(.)$ is a penalty function and $\lambda \in \mathbb{R}$ is the regularization parameter. As the goal is to obtain a sparse estimation of the vector of parameters $\boldsymbol{\beta}$, a natural choice for the penalty function is to use the so-called ℓ_0 norm ($\|.\|_0$), which corresponds to the number of non-vanishing elements of a vector:

$$\begin{array}{rccl} \operatorname{pen}_\lambda & : & \mathbb{R} & \mapsto & \{0,\lambda\} \\ & & x & \mapsto & \begin{cases} \operatorname{pen}_\lambda(x) = \lambda & \text{if } x \neq 0 \\ \operatorname{pen}_\lambda(x) = 0 & \text{else} \end{cases}, \end{array} \tag{2}$$

which induces $\sum_{p=1}^{P} \operatorname{pen}_\lambda(\beta_p) = \lambda\|\boldsymbol{\beta}\|_0$. For example, when λ =1, we get the Akaike Information Criterion (AIC) (Akaike, 1974) and when $\lambda = \frac{\log(N)}{2}$, we get the Bayesian Information Criterion (BIC) (Schwarz, 1978).

Many different penalties can be found in the literature. Solving this problem with $\|.\|_0$ as part of the penalty is an NP-hard problem (Fan and Lv, 2010; Natarajan, 1995). It cannot be used in practice when P becomes large, even when it is employed with some search strategy like forward regression, stepwise regression (Hocking, 1976) and genetic algorithms (Koza *et al.*, 1999). Donoho and Elad (2003) showed that relaxing $\|.\|_0$ to norm $\|.\|_1$ ends, under some assumptions, to the same estimation. This result encourages the use of a wide range of penalties based on different norms. For example, the case where $\operatorname{pen}_\lambda(\beta_p) = \lambda|\beta_p|$ is the lasso estimator (Tibshirani, 1996) [or equivalently Basis Pursuit Denoising (Chen *et al.*, 2001)] whereas $\operatorname{pen}_\lambda(\beta_p) = \lambda\beta_p^2$ leads to the Ridge estimator (Hoerl and Kennard, 1970). Nevertheless, the penalty term induces variable selection only if:

$$\min_{x\geq 0}\left(\frac{\mathrm{d}\operatorname{pen}_\lambda(x)}{\mathrm{d}x} + x\right) > 0. \tag{3}$$

Equation (3) explains why the lasso regression allows for variable selection, while the Ridge regression does not. The lasso regression is, however, known to lead to a biased estimate (Zou, 2006). The Smoothly Clipped Absolute Deviation (SCAD) (Fan, 1997), Minimax Concave Penalty (Zhang, 2010) or adaptive lasso (Zou, 2006) penalties all address this problem. The popularity of such variable selection methods is linked to fast algorithms like Least-Angle Regression Selection (Efron *et al.*, 2004), coordinate descent or Penalized Linear Unbiased Selection (Zhang, 2010).

Nevertheless, the goal of identifying the correct support of the regression is complicated and the reason why variable selection methods fail to select the set of non-zero variables $\mathcal{S}$ can be summarized in two words: linear correlation. Choosing the lasso regression as a special case, Zhao and Yu (2006) stated that if an irrelevant predictor is highly correlated with the predictors in the true model, lasso may not be able to distinguish it from the true predictors with any amount of data and any amount of regularization. Zhao and Yu (2006) [and simultaneously Zou (2006)] found an almost necessary and sufficient condition for lasso sign consistency (i.e. selecting the non-zero variables with the correct sign). This condition is known as 'irrepresentable condition':

$$|\mathbf{X}'_{\backslash\mathcal{S}}\mathbf{X}_{\mathcal{S}}(\mathbf{X}'_{\mathcal{S}}\mathbf{X}_{\mathcal{S}})^{-1}\operatorname{sgn}(\boldsymbol{\beta}_{\mathcal{S}})| < 1, \tag{4}$$

where $\mathbf{X}_{\mathcal{S}} = (x_{ij})_{i,j\in\mathcal{S}^2}$, $\mathbf{X}_{\backslash\mathcal{S}} = (x_{ij})_{i,j\neq\mathcal{S}^2}$, $\boldsymbol{\beta}_{\mathcal{S}} = (\beta_p)_{p\in\mathcal{S}}$. In other words, when $\operatorname{sgn}(\boldsymbol{\beta}_{\mathcal{S}}) = 1$, this can be seen as the regression of each variable, which is not in $\mathcal{S}$ over the variables, which are in $\mathcal{S}$. As all variables in the matrix $\mathbf{X}$ are centered, the absolute sum of the regression parameters should be smaller than 1 to satisfy this 'irrepresentable condition'.

Facing this issue, existing variable selection methods can be split into two categories:

- those which are 'regularized' and try to give similar coefficients to correlated variables [e.g. elastic net (Zou and Hastie, 2005)],
- those which are not 'regularized' and pick up one variable among a set of correlated variables [e.g. the lasso (Tibshirani, 1996)].

The former group can further be split into methods in which groups of correlation are known, such as the group lasso (Friedman *et al.*, 2010a; Yuan and Lin, 2006) and those in which groups are not known as in the elastic net (Zou and Hastie, 2005). The latter combines the ℓ_1 and the ℓ_2 norm and takes advantage of both. Non-regularized methods will select some co-variables among a group of correlated variables while regularized methods will select all variables in the same group with similar coefficients.

The main idea of our algorithm is to consider that any observed value of a group of linearly correlated variables of the $\mathbf{X}$ matrix is the independent realization of a given random function. This common random function is then used to perturb the observed values of the relevant correlated variables. Strictly speaking, the use of noise to determine the informative variables is not a new idea. For example, it has been shown that adding random pseudo-variables decreases over-fitting (Wu *et al.*, 2007). In the case where $P > N$, the pseudo-variables are generated either with a standard normal distribution $\mathcal{N}(0,1)$ or by using permutations on the matrix $\mathbf{X}$ (Wu *et al.*, 2007). Another approach consists of adding noise to the response variable and leads to similar results (Luo *et al.*, 2006). The rationale of this method is based on the work of Cook and Stefanski (1994), which introduces the simulation-based algorithm SIMEX (Cook and Stefanski, 1994). Adding noise to the matrix $\mathbf{X}$ has already been used in the context of microarrays (Chen *et al.*, 2007). Simsel (Eklund and Zwanzig, 2012) is an algorithm that both adds noise to variables and uses random pseudo-variables. One new and inspiring approach is stability selection (Meinshausen and Bühlmann, 2010) in which the variable selection method is applied on sub-samples, and informative variables are defined as variables which have a high probability of being selected. Bootstrapping has been applied to the lasso on both the response variable and the matrix $\mathbf{X}$ with better results in the former case (Bach *et al.*, 2008). A random lasso, in which variables are weighted with random weights, has also been introduced (Wang *et al.*, 2011).

In this article, following the idea of using simulation to enhance the variable selection methods, we propose the SelectBoost algorithm. Unlike other algorithms reviewed above, it takes into account the correlation structure of the data. Furthermore, our algorithm is motivated by the fact that in the case of non-regularized variable selection methods, if a group contains variables that are highly correlated together, one of them will be chosen with precision.

2 Materials and methods

The SelectBoost algorithm has been designed in a general framework in order to avoid to select non-predictive correlated features. The main goal is to improve the predictive positive value (PPV), i.e. the proportion of selected variables which truly belong to $\mathcal{S}$.

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

2.1 Generate new perturbed design matrix

As we assume that the variables are centered and that $\|\mathbf{x}_{p.}\|^2 = 1$ for $p = 1, \ldots, P$, we know that $\mathbf{x}_{p.} \in \mathcal{S}^{N-2}$. Indeed, the normalization puts the variables on the unit sphere $\mathcal{S}^{N-1}$. The process of centering can be seen as a projection on the hyperplane $\mathcal{H}^{N-1}$ with the unit vector as normal vector. Moreover, the intersection between $\mathcal{H}^{N-1}$ and $\mathcal{S}^{N-1}$ is $\mathcal{S}^{N-2}$. We further define the following isomorphism:

$$\begin{array}{rccc} \phi & : & \mathcal{H}^{N-1} & \rightarrow & \mathbb{R}^{N-1} \\ & & \mathbf{h}_n & \mapsto & \phi(\mathbf{h}_n) = \mathbf{f}_n \quad n = 1, \ldots, N-1, \end{array} \tag{5}$$

where $\{\mathbf{h}_n\}_{n=1,\ldots,N-1}$ is an orthogonal base of $\mathcal{H}^{N-1}$ and $\{\mathbf{f}_n\}_{n=1,\ldots,N-1}$ is the canonical base of $\mathbb{R}^{N-1}$. We define:

$$\mathbf{h}_n = \frac{\sum_{i=1}^{n} \mathbf{e}_i - n\mathbf{e}_{n+1}}{\|\sum_{i=1}^{n} \mathbf{e}_i - n\mathbf{e}_{n+1}\|},$$

with $\{\mathbf{e}_n\}_{n=1,\ldots,N}$ the canonical base of $\mathbb{R}^N$. Note that $\phi(\mathcal{S}^{N-2}) = \mathcal{S}^{N-2}$, and that is why we can work in $\mathbb{R}^{N-1}$ and then return in $\mathbb{R}^N$.

Here, we make the assumption that a group of correlated variables are independent realizations of the same multivariate Gaussian distribution. As the variables are normalized with respect to the ℓ_2 norm, we will use the von Mises–Fisher distribution (Sra, 2012) in $\mathbb{R}^{N-1}$ thanks to the isomorphism ϕ in order to generate new perturbed design matrix. The probability density function of the von Mises–Fisher distribution for the random P-dimensional unit vector $\mathbf{x}$ is given by:

$$f_P(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \tilde{K}_P(\kappa) \exp(\kappa \boldsymbol{\mu}' \mathbf{x}),$$

where $\kappa \geq 0$, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_P)'$, $\|\boldsymbol{\mu}\|_2 = 1$, and the normalization constant $\tilde{K}_P(\kappa)$ is equal to:

$$\tilde{K}_P(\kappa) = \frac{\kappa^{P/2-1}}{(2\pi)^{P/2} I_{P/2-1}(\kappa)},$$

where I_ν denotes the modified Bessel function of the first kind and order ν (Abramowitz and Stegun, 1972). We denote by $\hat{\boldsymbol{\mu}}$ and $\hat{\kappa}$ the maximum likelihood estimators of the $\boldsymbol{\mu}$ and κ parameters.

The multivariate Gaussian distribution assumption is not restrictive. As long as the group of correlated variables is independent realizations of the same distribution, the SelectBoost algorithm can be applied: either directly to assess the stability of the selected variables with perturbed datasets with an increasing noise level, which is the core idea behind the SelectBoost algorithm, or after replacing the von Mises–Fisher distribution with a more relevant one.

2.2 The SelectBoost algorithm

To use the SelectBoost algorithm, we need a grouping method gr_{c_0} depending on a user-provided constant $0 \leq c_0 \leq 1$. This constant determines the strength of the grouping effect. The grouping method maps each variable index $1, \ldots, P$ to an element of $\mathcal{P}(\{1, \ldots, P\})$ [with $\mathcal{P}(S)$ the powerset of the set S, i.e. the set which contains all the subsets of S]. Concretely, $gr_{c_0}(p)$ is the set of all variables, which are considered to be linked to the variable $\mathbf{x}_p$ and $\mathbf{X}_{gr_{c_0}(p)}$ is the submatrix of $\mathbf{X}$ containing the columns which indices are in $gr_{c_0}(p)$. We impose the following constraints to the grouping function:

$$\forall p \in \{1, \ldots P\} : gr_1(p) = \{p\} \quad \text{and} \quad gr_0(p) = \{1, \ldots P\}. \tag{6}$$

Furthermore, we need to have a selection method:

$$\text{select} \quad : \quad \mathbb{R}^{N \times P} \times \mathbb{R}^N \quad \rightarrow \quad \{0,1\}^P,$$

which maps the design matrix $\mathbf{X}$ and the response variable $\mathbf{y}$ to a 0–1 vector of length P with 1 at position p if the method selects the variable p and 0 otherwise. We then use the von Mises–Fisher distribution to generate replacement of the original variables by some simulations (see Algorithm 1) to create B new design matrices $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(B)}$. The SelectBoost algorithm then applies the variable selection method *select* to each of these matrices and returns a vector of length P with the frequency of apparition of each variable. The frequency of apparition of variable $\mathbf{x}_{p.}$, noted ζ_p is assumed to be an estimator of the probability $\mathbb{P}(\mathbf{x}_{p.} \in \mathcal{S})$ for this variable to be in $\mathcal{S}$. The choice of c_0 is crucial. On the one hand, when this constant is too large, the model is not perturbed enough. On the other hand, when this constant is too small, variables are chosen at random.

Algorithm 1 Pseudo-code for the SelectBoost algorithm

Require: gr_{c_0}, select, B, c_0 $\boldsymbol{\zeta} \leftarrow 0_P$
  for $b = 1, \ldots, B$ **do**
    $\mathbf{X}^{(b)} \leftarrow \mathbf{X}$
    for $p = 1, \ldots, P$ **do**
      $\mathbf{x}_{p.}^{(b)} \leftarrow \phi^{-1}(\text{random}-\text{vMF}(\hat{\boldsymbol{\mu}}(\phi(\mathbf{X}_{gr_{c_0}(p)})), \hat{\kappa}(\phi(\mathbf{X}_{gr_{c_0}(p)}))))$
    end for
    $\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta} + \text{select}(\mathbf{X}^{(b)}, \mathbf{y})$
  end for
  $\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta}/B$

The SelectBoost algorithm returns the vector $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_P)'$. Each of these values has to be compared to a threshold $\zeta_{\min}$ to determine which variables are selected: we choose to select a variable p if $\zeta_p \geq \zeta_{\min}$. The simulation study showed that the choice of the threshold is critical and the algorithm can be improved if we enforce that the ζ_p values—as functions of c_0—are non-increasing, see Figure 1 bottom. This additional requirement makes sense: the more variables the resampling process involves—with smaller c_0— the less a given variable will be selected.

2.3 Choosing the parameters of the algorithm

We first have to choose the grouping function. One of the simplest ways to define a grouping function gr_{c_0} is the following:

$$gr_{c_0}(p) = \{q \in \{1, \ldots, P\} \,|\, | < \mathbf{x}_{p.}, \mathbf{x}_{q.} > | \geq c_0\}. \tag{7}$$

In other words, the correlation group of the variable p is determined by variables whose correlation with $\mathbf{x}_{p.}$ is at least c_0. In another way, the structure of correlation may further be taken into account using graph community clustering. Let $\mathbf{C}$ be the correlation matrix of matrix $\mathbf{X}$. Let define $\check{\mathbf{C}}$ as follows:

$$\check{c}_{ij} = \begin{cases} |\check{c}_{ij}| & \text{if} \quad |\check{c}_{ij}| > c_0 \quad \text{and} \quad i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

Then, we apply a community clustering algorithm on the undirected network with weighted adjacency matrix defined by $\check{\mathbf{C}}$. Using a graph community clustering algorithm is helpful with large datasets while still clustering similar variables together. For instance, the fast greedy modularity optimization algorithm for finding community structure (Clauset *et al.*, 2004) runs in essentially linear time for many real-world networks given that they are sparse and hierarchical.

Once the grouping function is chosen, we have to choose parameter c_0. Due to the constraints in Equation (6), the SelectBoost algorithm results in the initial variable selection method when $c_0 = 1$. As we will show in the next section, the smaller the parameter c_0, the higher the precision of the resulting selected variables. On the other hand, it is obvious that the probability of choosing none of the variables (i.e. resulting in the choice of an empty set) increases as the parameter c_0 decreases. In the perspective of experimental planning, the choice of c_0 should result of a compromise between precision and proportion of active identified variables. Hence, the c_0 parameter can be used to introduce a confidence index γ_p related to the variable $\mathbf{x}_{p.}$:

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

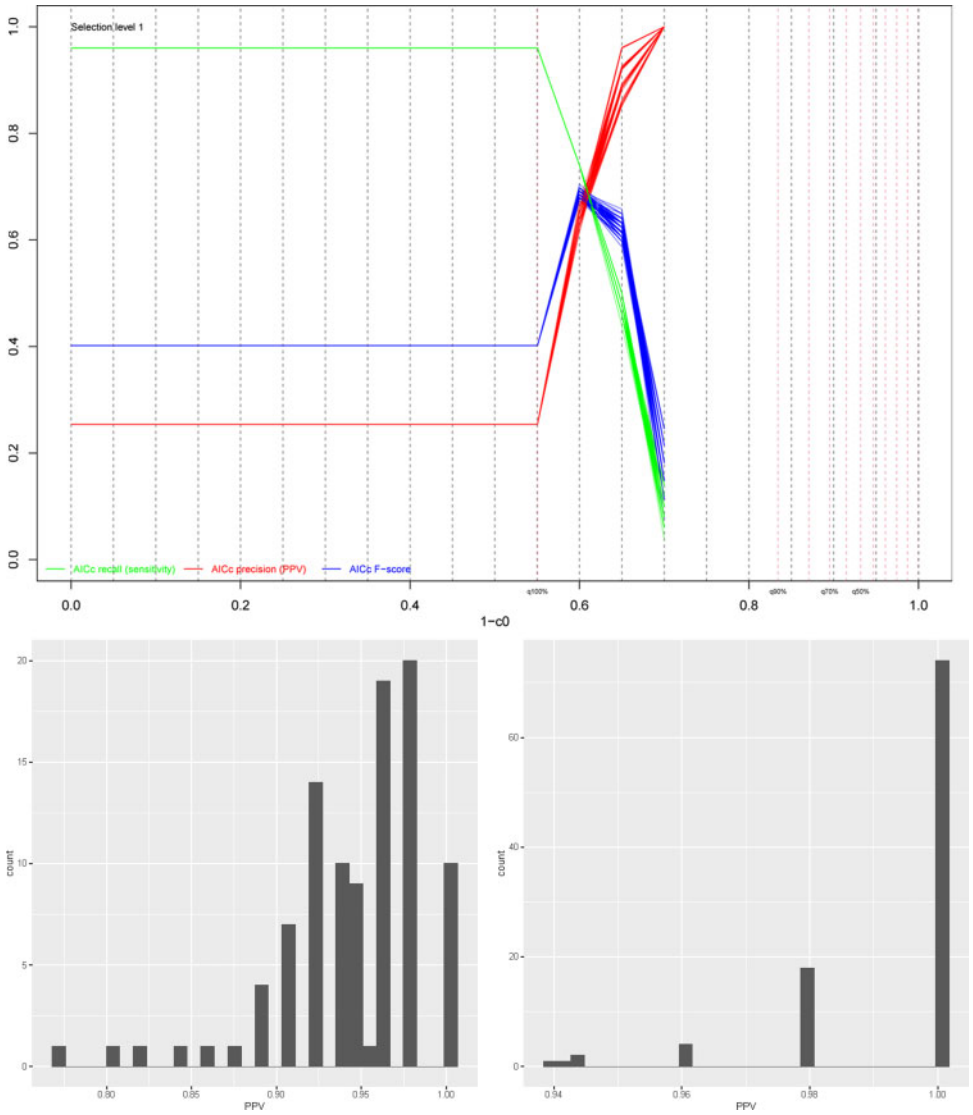

Fig. 1. Top: evolution of the recall, PPV and *F*-score as a function of $1 - c_0$ for LASSO-based SelectBoost and AICc model selection criterion for Type1 simulated data with a non-increasing post-processing step and a threshold $\zeta_{\min} = 1$. If $c_0 \leq 0.25$ models are empty. Bottom: the distribution of the PPV for a 0.25 threshold and $c_0 = \text{mean}(q_{90}, q_{100})$ for SPLS-based SelectBoost, Type1 data and raw SelectBoost (left) or SelectBoost with a non-increasing post-processing step (right)

$$\gamma_p = 1 - \min_{\mathbf{x}_{p\cdot} \in \hat{S}c_0} c_0, \text{hence} \quad 0 \leq \gamma_p \leq 1. \tag{8}$$

3 Numerical studies

We benchmarked the algorithm with a large simulation study with four data generation processes and three real datasets (Table 1). Generated datasets are available upon request and real datasets are available either upon request or online.

1. Simulation with 1000 variables and one linear response. A cluster of 50 variables is linked to the response.
2. Simulation with 1000 variables and one binary response. A cluster of 50 variables is linked to the response.
3. Data are 200 uncorrelated ('unlinked') single nucleotide polymorphisms (SNPs) with simulated genotypes, in which the first 20 of them affect the outcome with three covariates; 400 observations;
4. Data are 100 uncorrelated ('unlinked') SNPs with simulated genotypes, in which the first 10 of them affect the outcome with two covariates; 750 observations.
5. The leukemia dataset (Golub *et al.*, 1999) is the preprocessed data of Dettling (2004) retrieved from the Supplementary Material accompanying Friedman *et al.* (2010b).
6. The Huntington dataset is a real dataset with 28 087 variables observed on 69 individuals. We first applied independent filtering and removed 10 370 variables. We applied the SelectBoost algorithm to 17 717 variables observed on 69 individuals.
7. The melanoma dataset is the GSE78220 dataset from Hugo *et al.* (2016).

For Types 1 and 2, the number of variables is 1000, and the number of observations is 100. The data are generated from a cluster simulation (Bair *et al.*, 2006; Bastien *et al.*, 2015). Only 50 first predictors are linked to the response Y and the last 950 variables are randomly generated from a standard normal distribution. For Example 3, the response variable is linear but was turned into a binary variable (+1 when $Y_i > 0$ and –1 when $Y_i < 0$).

Examples 1 and 3 are linear regression examples whereas 2, 4, 5, 6 and 7 are logistic regression ones, for which, we will assume a logistic model with a binary response variable (Peng *et al.*, 2002).

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

Table 1. Summary of the types of datasets used to benchmark the SelectBoost algorithm

Name	Data	Individuals	Variables
Type1	Simulated	100	1000
Type2	Simulated	100	1000
Type3	Simulated	400	203
Type4	Simulated	750	102
Leukemia	Observed	72	3571
Huntington	Observed	69	17 717
Melanoma	Observed	28	25 268

We provide results for 12 different settings based on 10 different models, see Supplementary Section S1 for more details.

1. Linear regression (seven types): SPLS [Chun and Keles (2010), with raw and bootstrap corrected coefficients], LASSO, adaptive LASSO, enet and adaptive enet with model choice based on information criteria (AICc, BIC, GCV, Cp), LASSO with model choice based on 5-fold cross-validation ($\lambda_{\min}$, λ_{1se}) and varbvs linear (Carbonetto and Stephens, 2012; Guan and Stephens, 2011; Zhou *et al.*, 2013).
2. Logistic regression (five types): logistic LASSO (glmnet based) with model choice based on 5-fold cross-validation, logistic LASSO (glmnet based) with model choice based on information criteria (AICc, BIC), varbvs binomial, SPLSda (Chun and Keles, 2010) and sgpls (Chun and Keles, 2010).

The SelectBoost algorithm is based on correlated resampling and hence random. We wanted to assess both the stability and performance of the algorithm. As a consequence, for the four types of simulated data, we focused both on what may be called a repeatability study (a given dataset was analyzed 100 times to estimate the variation only due to the fact that the algorithm is random) and a reproducibility study (100 different datasets were generated and analyzed to estimate the variability due to both data simulation—from the same data generation—and the fact that the algorithm is random).

The repeatability issue raised was raised, for instance by Boulesteix (2014) and Magnanensi *et al.* (2017) for PLS models. For those models, random split cross-validation is known to have poor repeatability. We used two types of grouping functions (either determined by variables whose correlation with x_p is at least c_0 -gdirect- or community clustering-based -gcc-).

The cost (memory and time) of the random generation step can be limited thanks to a sparse correlated resampling feature. The remaining cost of the algorithm is $B \times Nc0 \times \text{Time}_1$ with B the number of resampling and $Nc0$ the number of c0 values that are investigated and Time_1 the time to fit the model once.

To demonstrate the performance of the SelectBoost method, we compared our method with stability selection (Meinshausen and Bühlmann, 2010) and with a naive version of our algorithm, naiveSelectBoost. The naiveSelectBoost algorithm works as follows: estimate β with any variable selection method then if $gr_{c_0}(p)$, as defined in Equation (7) e.g. is not reduced to p, shrink to 0. The naiveSelectBoost algorithm is similar to the SelectBoost algorithm, except that it does not take into account the error, which is made choosing at random a variable among a set of correlated variables.

We use four indicators to evaluate the abilities of our method on simulated data. We define:

- recall as the ratio of the number of correctly identified variables (i.e. $\hat{\beta}_i \neq 0$ and $\beta_i \neq 0$) over the number of variables that should have been discovered (i.e. $\beta_i \neq 0$).
- precision as the ratio of correctly identified variables (i.e. $\hat{\beta}_i \neq 0$ and $\beta_i \neq 0$) over the number of identified variables (i.e. $\hat{\beta}_i \neq 0$).
- *F*-score as the following ratio:

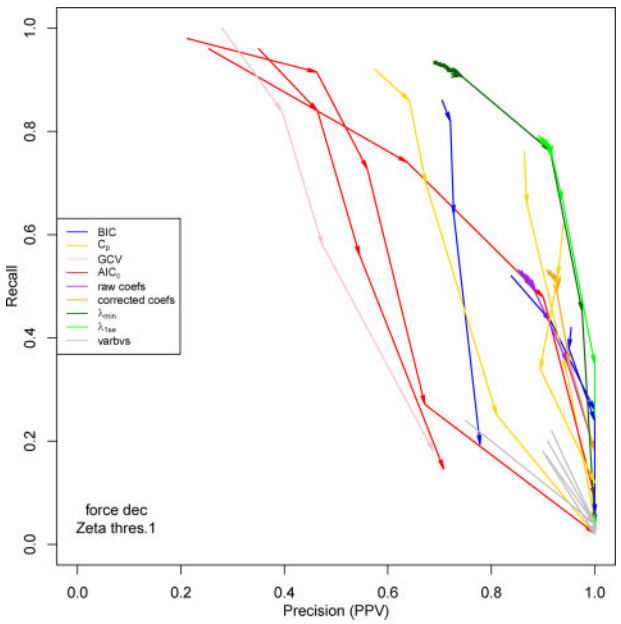

Fig. 2. Recall-precision curve. All models and criteria non-increasing SelectBoost. Type 1 data. Direct grouping. A total of 100 different datasets. $\zeta_{\min} = 1$

$$2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}.$$

- selection as the average number of identified variables (i.e. $\hat{\beta}_i \neq 0$).

Note that our interest is focused on precision, as our goal is to select reliable variables. As stated before, when c_0 is decreasing toward zero, we expect a profit in precision and a decrease in recall. We also compute the *F*-score, which combines both recall and precision. As an improvement of precision comes with a decrease in the number of identified variables, the best method is the one with the highest precision for a given level of selection.

3.1 Results of the numerical studies

We show the evolution of the four criteria (recall, precision, *F*-score and selection) with regards to the decrease of c_0. When $c_0 = 1$, the SelectBoost algorithm is equivalent to the initial variable selection method. We introduce a post-processing step to enforce that, for a given variable, the proportion of selection is non-increasing. It is the expected behavior since the correlated resampling is not meant to increase the probability of selection for a variable. Such an increase may happen for small c_0 values when a variable that is not linked with the response is mixed with a variable that is linked to the response. For all the simulations, this post-processing step increases the PPV of the SelectBoost algorithm, see Figure 1. As our primary focus is PPV, we recommend the use of this post-processing step. More details can be found in the Supplementary Graphs S7–S174.

We created precision-recall plots to display the effects of the algorithm on the performance of all the models and criteria used for a given dataset. Identical model fitting criteria share the same colors. The arrows point toward decreasing c_0 values. Direct grouping and community grouping lead to similar results, Figure 2 and Supplementary Figures S1, S3 and S4. These Figures also show that the results for a single dataset repeated 100 times are similar to results for 100 different datasets. The Zoom *l* sequence, which is a 10-step regular grid from the $q_{100\%}$ quantile—the maximum value—to $q_{90\%}$—the quantile of order 0.9—achieves high PPV, Supplementary Figure S299.

Supplementary Figure S5 displays an example of raw SelectBoost (without the non-increasing post-processing step) for direct grouping and 100 different datasets that should be compared to Figure 2.

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

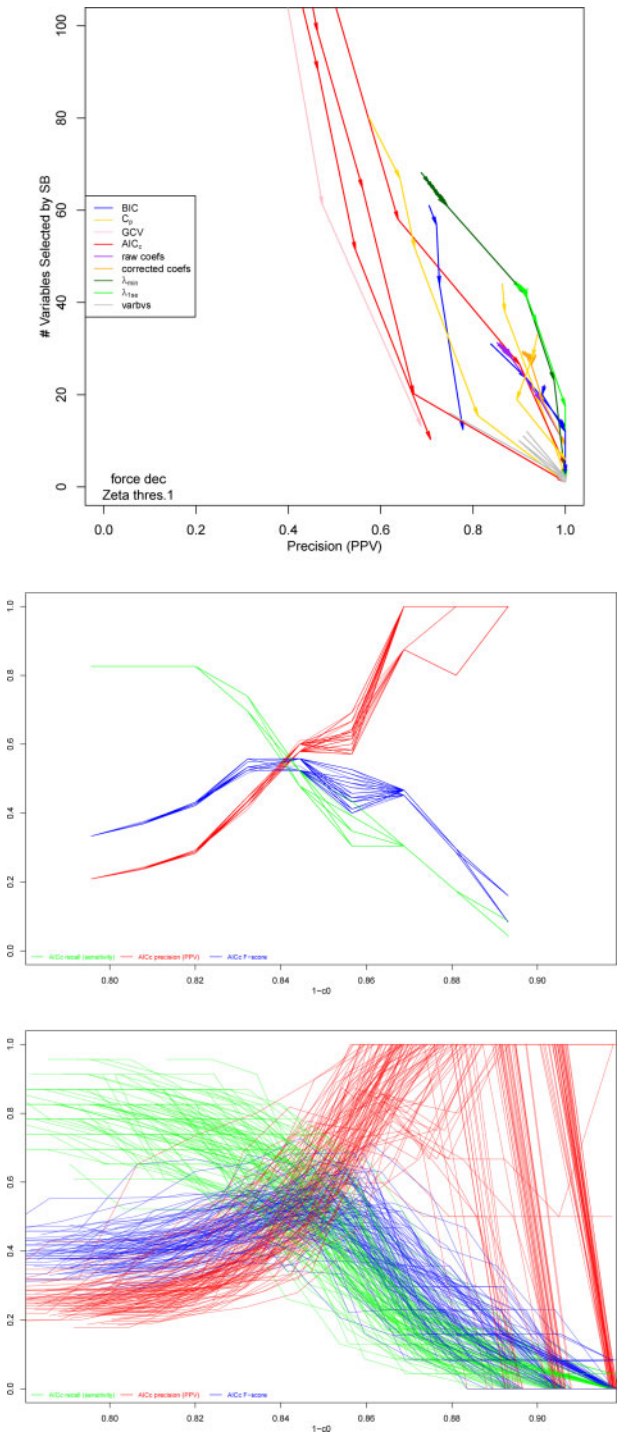

Fig. 3. Top: The average number of identified variables is plotted as a function of the proportion of correctly identified variables for Type1 simulated data and all models. Middle and bottom: effect of the SelectBoost algorithm wrt $1 - c_0$ for adaptive elastic net and AICc model selection criterion with c_0 in the range $[q_{90\%}; q_{100\%}]$ for 100 different (middle, reproducibility) or 100 identical (bottom, repeatability) Type3 simulated data with a non-increasing post-processing step and a threshold $\zeta_{min} = 1$. Only results for non-empty models are shown

This effect is even stronger smaller values for the ζ_{min} threshold. The non-increasing post-processing step greatly improves the results of the algorithm and leads to monotonic relationships between the recall, the precision and the c_0 value. PPV benefit less from smaller values for the ζ_{min} threshold, Supplementary Figure S6.

All the results of the simulation study showed the good performance and stability of the algorithm, which we then applied once to each of the real datasets. The results of the gdirect and gcc based SelectBoost are similar, the gcc based being a bit more time consuming than the gdirect one. In the following of this article, we reported results and figures for gdirect-based SelectBoost.

According to our simulation studies, Supplementary Graphs S7–S174, one should choose c_0 between $q_{90\%}$ and $q_{100\%}$, see Figure 1 top. In our simulation studies, we used an 11-steps c_0 sequence, but, according to our results, it could be limited to 6 steps [from mean$(q_{90\%}, q_{100\%})$ to $q_{100\%}$] for the biggest datasets. The value of B should not be lower than 10. $B = 50$ or $B = 100$ will provide more stable results. As a consequence, the minimal time cost of the SelectBoost algorithm will be 60 times the time cost of the regular model fit, which could be afforded in almost every case. The parallel processing support of the SelectBoost package can help to reduce this time. Hence, the SelectBoost seems feasible with most of the datasets and even omics datasets as we did in our simulation study with the three real datasets.

Hence, to assess the performance of the SelectBoost algorithm, we performed comprehensive numerical studies. As stated before, the SelectBoost algorithm can be applied to any existing variable selection method.

Figure 1 top shows the result for the lasso selection with a penalty parameter chosen using information criteria for Type 1 datasets. In this example, we improve the precision up to 1. Moreover, as shown by Figures 1 and 3, the proportion of models, for which the precision reaches one, increases with the decrease of c_0. The F-score increases, remains either stable or shows a small decrease indicating that the increase of PPV compensates the loss in recall.

In the previous section, we mentioned the possibility of using SelectBoost to obtain a confidence index, corresponding to one minus the lowest c_0 for which a variable is selected. For each c_0, we plotted the average number of selected variables as a function of the proportion of correctly identified variables (Fig. 3 and Supplementary Figs S300–S304). As expected, the proportion of correctly identified variables increases with the increase of the confidence index and with the decrease of the average number of identified variables. Therefore, the proportion of non-predictive features decreases with the increase of the confidence index.

The SelectBoost algorithm shows its superiority over the naive SelectBoost algorithm. The error made when choosing a variable randomly among a set of correlated variables leads to more incorrect choices of variables. While the intensive simulation of our algorithm allows taking into account this error, the naiveSelectBoost does not.

Finally, we compare the SelectBoost algorithm with stability selection. Stability selection uses a resampling algorithm to determine which of the variables included in the model are robust. In our simulation, stability selection shows performance with high precision but also low recall. Moreover, in contrast to the SelectBoost algorithm, stability selection does not allow to choose a convenient precision-PPV trade-off.

The timings of the algorithm can be found on Supplementary Figures S175–S222.

4 Application to three real datasets

We applied our algorithm to three real datasets. We studied, with respect to the threshold, the number of non-zero variables, the number of variables selected by SelectBoost and their ratio. We found results that were concordant with those of the simulated datasets. Figure 4 displays those results for a SGPLS-based SelectBoost of the Leukemia dataset with a 0.25 threshold ($\zeta_{min} = 0.25$). See Supplementary Figures S247–S298.

We report the results for the RNA-Seq dataset providing mRNA expressions from Huntington's disease and neurologically normal individuals. This dataset was downloaded from the GEO database under accession number GSE64810 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi? acc=GSE64810). This dataset contains 20 Huntington's disease cases and 49 neurologically normal controls and includes 28 087 genes as explanatory variables. An independent

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

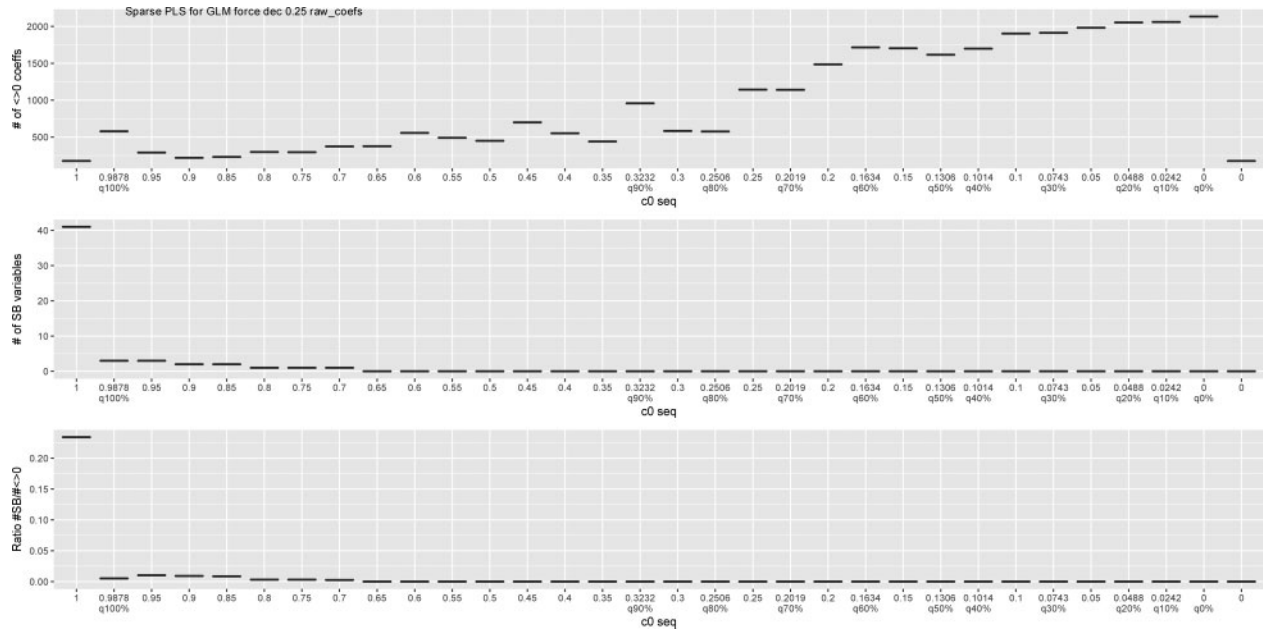

Fig. 4. % of non-zero coefficients wrt to c_0 for SGPLS-based SelectBoost models of the leukemia datasets and threshold $\zeta_{\min} = 0.25$

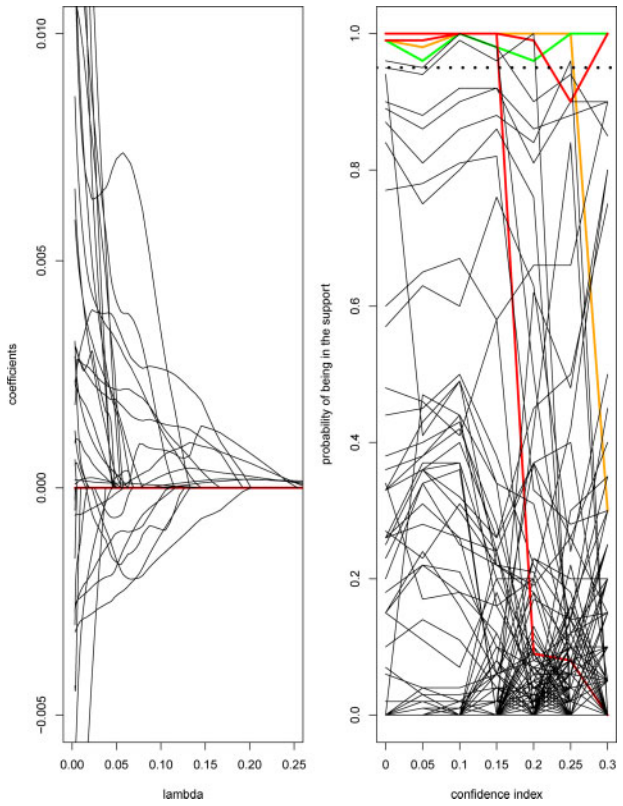

Fig. 5. Colors: the green is for the most reliable variables selected by the SelectBoost algorithm [confidence index of 0.3; orange is for intermediate confidence (0.25) and red for low confidence (0.15)]. Left: evolution of the coefficients in the lasso regression when the regularization parameter λ is varying. For the λ range shown, the red, orange and green lines stick to zero. Right: evolution of the probability of being in the support of the regression when the confidence index is varying. The dotted line represents the 0.95 threshold

filtering (Bourgon *et al.*, 2010) preprocessing step was first performed using data-based filtering for replicated high-throughput transcriptome sequencing experiments (Rau *et al.*, 2013). Then, we applied the lasso selection method to this reduced dataset (see Fig. 5 left for the whole path of the solution). We used cross-validation to choose the appropriate level of penalization [i.e. the λ parameter in Equation (3)].

We then applied our SelectBoost algorithm on the lasso method with penalty parameter chosen by cross-validation. We use a range for the c_0 parameter starting from 1 to 0.7 with steps of 0.05, which corresponds to a confidence index from 0 to 0.3. For each step, the probability of being included in the support $\mathcal{S}$ was calculated with 200 simulations as described in Algorithm 1. We set the threshold of being in the support to 0.95 to avoid numerical instability. We classify the selected variables into three categories: those that are identified for each confidence index from 0 to 0.15 (red), those identified from 0 to 0.25 (orange) and those identified from 0 to 0.3 (green). The last category contains the most reliable variables selected by the SelectBoost algorithm because these variables are identified from low to high confidence index.

With the lasso selection method, 15 variables were selected. Among them, four genes were identified by SelectBoost into the three different categories of confidence index (see Fig. 5 right): two genes for low confidence (red) (ANXA3 and INTS12), one gene for intermediate confidence (orange) (NUB1) and one gene for high confidence (green) (PUS3).

The interesting point, in these three examples, is that the identified variables are neither the first variables selected by the lasso nor the variables with the highest coefficients (see Fig. 5 left). This result demonstrates that our algorithm can be advantageous to select variables with high confidence and not just to select variables with the highest coefficients.

Finally, we decided to assess the differential expression of these genes between patients and controls, using the `limma` package (Linear Models for Microarray and RNA-Seq Data) (Ritchie *et al.*, 2015). The four identified genes are significantly down-expressed by neurologically healthy controls confirming the result of a logistic model with these four genes.

5 Robust reverse-engineering of networks

Sparsity is a well-known feature of most biological networks (Barabási *et al.*, 2003). An actor can only be regulated by a small number of other actors, whereas it may regulate any number of other actors. Hence, variable selection methods, such as the lasso, ensure that sparsity feature and are often core components of most

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

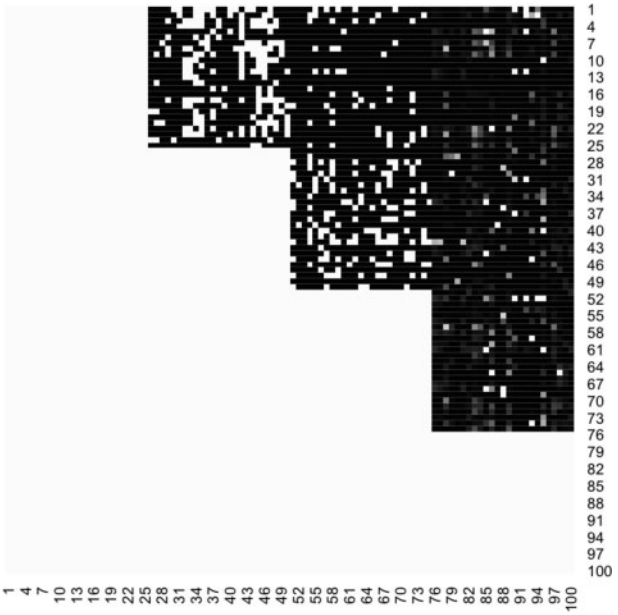

Fig. 6. Post-inference analysis of an inferred cascade network. Dark values are tantamount to low confidence. Bright values are tantamount to high confidence. Confidence ranges from 0 (lowest) to 1 (highest). The lower triangular part of the matrix is an area with the highest confidence (1) since we know—and assume so in the model—that for cascade networks those links must be =0

of the biological network reverse-engineering tools. As a consequence, we propose to apply the SelectBoost algorithm in two different ways in order to improve the biological network reverse-engineering: as a post-processing step after the inference was made or during the inference itself in order to select the most stable predictors for each node in the network. When used as a post-processing step, one can assess for any of the inferred links between the actors of the network, its confidence index against correlated resampling of the predictors. When used during the inference step, one can infer a model that is only built with links with a high enough confidence index. The former is implemented in the SelectBoost package as a new method for the Cascade package (Jung *et al*, 2014) . The latter is implemented in the new Patterns CRAN package as a dedicated fitting function and is especially useful when trying to find targets for biological intervention that are strongly related to markers of some diseases through the reverse-engineered network and useful and reliable links.

We benchmarked those two uses of the algorithm with a particular type of biological networks that we have been using for several years: cascade networks (Vallat *et al*., 2013).

For the post-inference processing, we first fit a model to a cascade network using the Cascade package inference function. Then, we compute confidence indices for the inferred links using the SelectBoost algorithm, more details, as well as the code, of the simulations can be found in the vignette of the package 'Towards Confidence Estimates in Cascade Networks using the SelectBoost Package', available at https://fbertran.github.io/SelectBoost/articles/confidence-indices-Cascade-networks.html. An example of those results is shown in Figure 6 with a cascade network for four time points and four groups of 25 actors.

For the use of the SelectBoost algorithm during the fitting step of a cascade network reverse-engineering, we used the Patterns package. Benchmark results were reported as sensitivity, positive predictive value and *F*-score, shown in Figure 7; the code, the simulation details and the remaining results are part of a vignette of the package 'Benchmarking the SelectBoost Package for Network Reverse Engineering', that is available at https://fbertran.github.io/SelectBoost/articles/benchmarking-SelectBoost-networks.html.

We created an unweighted or a weighted version of the algorithm. The weighted version of the algorithm enables the user to include weights in the model, which means to favor or disfavor some links between the actors, in order, for instance, to take into account biological knowledge.

The results shown in Figure 7 of the simulation study are a comparison to a standard set up for stability selection and regular lasso both for an unweighted version of the algorithms and a highly correctly weighted version of the same algorithms.

By highly correctly weighted, we mean that we included influential weights in the model accordingly to the links that existed in the network that was used for data simulation. This network was randomized from one simulation to another. This weighted setting was used to determine if including correct biological knowledge would help the reverse-engineering algorithm to retrieve the correct network. If correct biological knowledge is included in the model, all three fitting functions lead to similar and outstanding results for the *F*-score criterion without even requiring the need to search for an optimal thresholding value as we had to do with the Cascade package.

For each simulated dataset, vertical dots are displayed to show the optimal threshold level that should be used to maximize the *F*-score. It is computed with respect to the actual values that are unknown for real datasets. Without weights, SelectBoost shrinks the range of optimal values when compared to the lasso or stability selection. With correct weights, none of the methods still requires to use a cut-off value to maximize *F*-score.

In an unweighted setting, the SelectBoost version of the fitting process shows better performance than stability selection and the lasso as long as the cut-off value is <0.4, which is about the double of the optimal thresholding value.

6 Conclusion

We introduce the SelectBoost algorithm that relies intensive computations to select variables with high precision (PPV). The user of SelectBoost can apply this algorithm to produce a confidence index or choose an appropriate precision-selection trade-off to select variables with high confidence and avoid selecting non-predictive features. The main idea behind our algorithm is to take into account the correlation structure of the data and thus use intensive computation to select reliable variables.

The choice of the threshold $\zeta_{\min}$ is critical since such a choice leads to two effects.

- With a high threshold value—nearing the maximum value of 1—: an increase of the PPV while limiting the decrease of the *F*-score.
- With a low or medium threshold value—nearing the mid value of 0.5—: an increase in recall while limiting the decrease of the *F*-score.

We will want the first property to retrieve the stable core of the predictors for models that are known to randomly choose between correlated variables, such as the lasso or adaptive lasso. . Whereas, we will want the second property for models that scarcely select variable, such as variable selection model using variational approximation methods for binary response (`varbvs`). A non-constant threshold should be also and investigated by those that would like to introduce corrections, for instance FDR-like, such as Holm–Bonferroni, in the variable selection process.

We prove the performance of our algorithm through simulation studies in various settings. To get the best results, we recommend the use of c_0 in the range of mean$(q_{90\%}, q_{100\%})$ to $q_{100\%}$ with the non-increasing post-processing step. It could be useful to decrease the lower bound to $q_{90\%}$ for the smallest datasets. The user should never use a c_0 value too close to the empty model zone to avoid a decrease in precision. We succeed in improving the PPV, whenever it was possible, of all the 12 selection methods with relative stability on recall and *F*-score. If the PPV was already nearing 1, then there is almost no negative effect on the PPV and recall when applying SelectBoost.

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

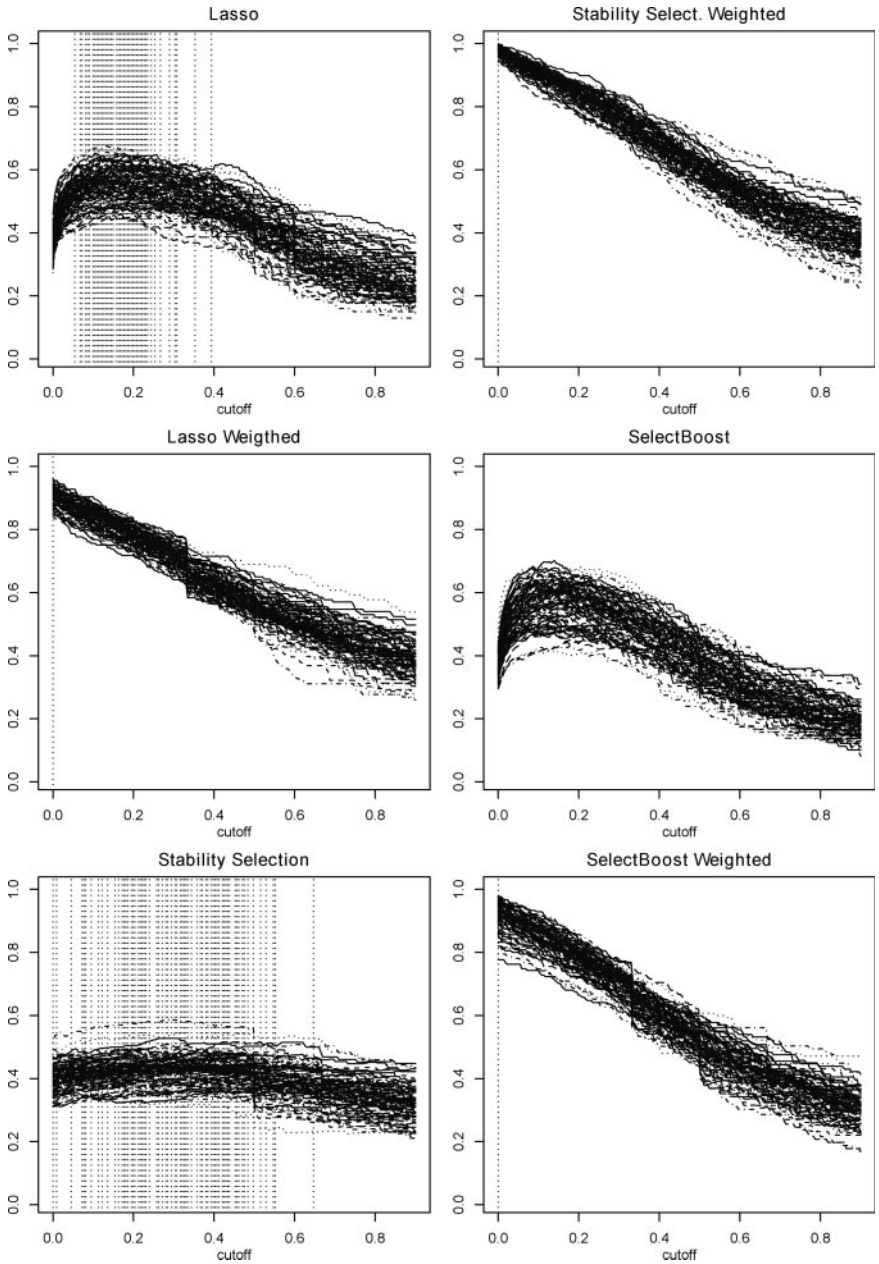

Fig. 7. *F*-score as a function of the thresholding value: if an inferred coefficient for the network is less than the thresholding value, then it is set to 0. The SelectBoost algorithm is compared to both stability selection and the regular lasso. The upper row displays results for the unweighted version of the algorithms, whereas the lower row displays results for their weighted counterparts

Our results open the perspective of a precision-selection trade-off which may be very useful in some situations where many regressions have to be made (e.g. network reverse-engineering with one regression made per node of the network). In such a context, our algorithm may even be used in an experimental design approach.

The application to three real datasets allowed us to show that the most reliable variables are not necessarily those with the highest coefficients. The SelectBoost algorithm is a powerful tool that can be used in every situation where reliable and robust variable selection has to be made.

Funding

This work was supported by grants from the Agence Nationale de la Recherche (ANR) [ANR-11-LABX-0070_TRANSPLANTEX]; the INSERM [UMR_S 1109]; the Institut Universitaire de France (IUF) and the MSD-Avenir grant AUTOGEN, all to S.B.; the European regional development fund (European Union) INTERREG V program (project number 3.2 TRIDIAG) to R.C. and S.B.; the Agence Nationale de la Recherche (ANR) [ANR-11-LABX-0055_IRMIA]; the CNRS [UMR 7501] to F.B. and M.M.-B.; and by the French HPC Center ROMEO [UR 201923174L] to F.B.

Conflict of Interest: none declared.

References

Abramowitz,M. and Stegun,I.A. (1972) *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Vol. **55**, National Bureau of Standards Applied Mathematics Series, Dover Publications Inc., New York.

Akaike,H. (1974) A new look at the statistical model identification. *IEEE Trans. Automat. Contr.*, **19**, 716–723.

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021

Bach,F.R. *et al.* (2008) Bolasso: model consistent lasso estimation through the bootstrap. In: *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland. pp. 33–40.
Bair,E. *et al.* (2006) Prediction by supervised principal components. *J. Am. Stat. Assoc.*, **101**, 119–137.
Barabási,A.L. *et al.* (2003) Emergence of scaling in complex networks. In: Bornholdt,S. and Schuster,H.G. (eds) *Handbook of Graphs and Networks: From the Genome to the Internet*. Wiley-VCH, Weinheim, pp. 69–84.
Bastien,P. *et al.* (2015) Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data. *Bioinformatics*, **31**, 397–404.
Boulesteix,A.-L. (2014) Accuracy estimation for PLS and related methods via resampling-based procedures. In: *PLS–14 Book of Abstracts, Paris, France*. pp. 13–14.
Bourgon,R. *et al.* (2010) Independent filtering increases detection power for high-throughput experiments. *Proc. Natl. Acad. Sci. USA*, **107**, 9546–9551.
Chen,L. *et al.* (2007) Noise-based feature perturbation as a selection method for microarray data. In: *Bioinformatics Research and Applications, Atlanta, GA, USA*. pp. 237–247.
Chen,S.S. *et al.* (2001) Atomic decomposition by basis pursuit. *SIAM Rev.*, **43**, 129–159.
Carbonetto,P. and Stephens,M. (2012) Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Anal.*, **7**, 73–108.
Chun,H. and Keles,S. (2010) Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *J. R. Stat. Soc. Series B Stat. Methodol.*, **72**, 3–25.
Clauset,A. *et al.* (2004) Finding community structure in very large networks. *Phys. Rev. E*, **70**, 066111.
Cook,J.R. and Stefanski,L.A. (1994) Simulation-extrapolation estimation in parametric measurement error models. *J. Am. Stat. Assoc.*, **89**, 1314–1328.
Dettling,M. (2004) BagBoosting for tumor classification with gene expression data. *Bioinformatics*, **20**, 3583–3593.
Donoho,D.L. and Elad,M. (2003) Optimally sparse representation in general (nonorthogonal) dictionaries via L1 minimization. *Proc. Natl. Acad. Sci. USA*, **100**, 2197–2202.
Efron,B. *et al.* (2004) Least angle regression. *Ann. Stat.*, **32**, 407–499.
Eklund,M. and Zwanzig,S. (2012) SimSel: a new simulation method for variable selection. *J. Stat. Comput. Simul.*, **82**, 515–527.
Fan,J. (1997) Comments on "Wavelets in statistics: a review" by A. Antoniadis. *Stat. Meth. Appl.*, **6**, 131–138.
Fan,J. and Li,R. (2006) Statistical challenges with high dimensionality: feature selection in knowledge discovery. In: *Proceedings International Congress of Mathematicitans*. **Vol. 3**, pp. 595–622, European Mathematical Society, Zúrich.
Fan,J. and Lv,J. (2010) A selective overview of variable selection in high dimensional feature space. *Stat. Sin.*, **20**, 101–148.
Friedman,J. *et al.* (2010a) A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv: 1001.0736*.
Friedman,J. *et al.* (2010b) Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.*, **33**, 1–22.
Golub,T.R. *et al.* (1999) Molecular classification -of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.
Guan,Y. and Stephens,M. (2011) Bayesian variable selection regression for genome-wide association studies and other large-scale problems. *Ann. Appl. Stat.*, **5**, 1780–1815.
Hocking,R.R. (1976) A Biometrics invited paper. The analysis and selection of variables in linear regression. *Biometrics*, **32**, 1–49.
Hoerl,A.E. and Kennard,R.W. (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55–67.
Hugo,W. *et al.* (2016) Genomic and transcriptomic features of response to anti-PD-1 therapy in metastatic melanoma. *Cell*, **165**, 35–44.
Jung,N. *et al.* (2014) Cascade: a R package to study, predict and simulate the diffusion of a signal through a temporal gene network. *Bioinformatics*, **30**, 571–573.
Koza,J.R. *et al.* (1999) *Genetic Programming as a Darwinian Invention Machine*. Springer, Heidelberg.
Lipshutz,R.J. *et al.* (1999) High density synthetic oligonucleotide arrays. *Nat. Genet.*, **21**, 20–24.
Luo,X. *et al.* (2006) Tuning variable selection procedures by adding noise. *Technometrics*, **48**, 165–175.
Magnanensi,J. *et al.* (2017) A new universal resample-stable bootstrap-based stopping criterion for PLS component construction. *Stat. Comput.*, **27**, 757–718.
Meinshausen,N. and Bühlmann,P. (2010) Stability selection. *J. R. Stat. Soc. Series B Stat. Methodol.*, **72**, 417–473.
Morgan,D. *et al.* (2019) A generalized framework for controlling FDR in gene regulatory network inference. *Bioinformatics*, **35**, 1026–1032.
Natarajan,B.K. (1995) Sparse approximate solutions to linear systems. *SIAM J. Comput.*, **24**, 227–234.
Peng,C.J. *et al.* (2002) An introduction to logistic regression analysis and reporting. *J. Educ. Res.*, **96**, 3–14.
Rau,A. *et al.* (2013) Data-based filtering for replicated high-throughput transcriptome sequencing experiments. *Bioinformatics*, **29**, 2146–2152.
Ritchie,M.E. *et al.* (2015) limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.*, **43**, e47.
Schwarz,G. (1978) Estimating the dimension of a model. *Ann. Stat.*, **6**, 461–464.
Segal,M.R. *et al.* (2003) Regression approaches for microarray data analysis. *J. Comput. Biol.*, **10**, 961–980.
Sra,S. (2012) A short note on parameter approximation for von Mises-Fisher distributions: and a fast implementation of I s (x). *Comput. Stat.*, **27**, 177–190.
Tibshirani,R. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B Methodol.*, **58**, 267–288.
Vallat,L. *et al.* (2013) Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proc. Natl. Acad. Sci. USA*, **110**, 459–464.
Wang,S. *et al.* (2011) Random lasso. *Ann. Appl. Stat.*, **5**, 468–485.
Wu,Y. *et al.* (2007) Controlling variable selection by the addition of pseudo-variables. *J. Am. Stat. Assoc.*, **102**, 235–243.
Yuan,M. and Lin,Y. (2006) Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Series B Stat. Methodol.*, **68**, 49–67.
Zhang,C. (2010) Nearly unbiased variable selection under minimax concave penalty. *Ann. Stat.*, **38**, 894–942.
Zhao,P. and Yu,B. (2006) On model selection consistency of lasso. *J. Mach. Learn. Res.*, **7**, 2541–2563.
Zhou,X. *et al.* (2013) Polygenic modeling with Bayesian sparse linear mixed models. *PLoS Genet.*, **9**, e1003264.
Zou,H. (2006) The adaptive lasso and its oracle properties. *J. Am. Stat. Assoc.*, **101**, 1418–1429.
Zou,H. and Hastie,T. (2005) Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Series B Stat. Methodol.*, **67**, 301–320.

Downloaded from https://academic.oup.com/bioinformatics/article/37/5/659/5918024 by Universite de Strasbourg France user on 28 October 2021