LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR STATISTIK

Riccardo De Bin

Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages *CoxBoost* and *mboost*

Technical Report Number 180, 2015
Department of Statistics
University of Munich

http://www.stat.uni-muenchen.de

Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages *CoxBoost* and *mboost*

Riccardo De Bin[*]

April 2, 2015

Abstract

Despite the limitations imposed by the proportional hazards assumption, the Cox model is probably the most popular statistical tool used to analyze survival data, thanks to its flexibility and ease of interpretation. For this reason, novel statistical/machine learning techniques are usually adapted to fit it, including boosting, an iterative technique originally developed in the machine learning community and later extended to the statistical field. The popularity of boosting has been further driven by the availability of user-friendly software such as the R packages *mboost* and *CoxBoost*, both of which allow the implementation of boosting in conjunction with the Cox model. Despite the common underlying boosting principles, these two packages use different techniques: the former is an adaption of the model-based boosting, while the latter adapts the likelihood-based boosting. Here we contrast these two boosting techniques as implemented in the R packages from an analytic point of view, and we examine the solutions there adopted to treat mandatory variables, i.e. variables that for some reasons must be included in the model. We explore the possibility of extending solutions currently only implemented in one package to the other. We illustrate the usefulness of these extensions through the application to two real data examples.

keywords: boosting, *CoxBoost*, Cox model, mandatory variables, *mboost*.

[*]debin@ibe.med.uni-muenchen.de - Department of Medical Informatics, Biometry and Epidemiology, University of Munich, Germany

1 Introduction

Among the iterative methods that have been largely exploited during the recent years in the statistical practice, a particular attention has been gained by the boosting: originally developed in the machine learning community (Schapire, 1990; Freund, 1995; Freund & Schapire, 1996), primarily to face classification problems, it has been successfully translated into the statistical field (Breiman, 1998; Friedman et al., 2000) and extended to any statistical problem, including regression and survival analysis. Thanks to its resistance to overfitting, it is particularly interesting in the construction of prediction models. Its iterative nature, moreover, allows straightforward adaptations to cope with high-dimensional data (Bühlmann & Yu, 2003; Bühlmann, 2006; Tutz & Binder, 2006; Binder & Schumacher, 2008). The so called component-wise boosting, in particular, satisfies the two most important needs for a procedure whose goal is to derive a prediction model from high-dimensional data: variable selection and shrinkage of the coefficients toward 0. These two aspects, in particular, are pursued simultaneously. Applied in a parametric framework, the basic idea of boosting is to provide estimates of the parameters by updating their values step by step: at each iteration, a weak estimator is fitted on a modified version of the data, with the goal of minimizing a pre-specified loss function. The obtained value provides a small contribution used to update the estimate of the parameter: the result of all the contributions is the final estimate. The boosting relies on two tuning parameters: a first parameter controls the "weakness" of the estimator and it is usually called *penalty* or *boosting step* (hereafter, we use the former definition). A second one, by far more influential, is related to the stopping criterion, i.e. specifies how many boosting iterations are performed. The latter parameter, in particular, plays an important role in avoiding overfitting, and, in case of component-wise version, it also controls the sparsity of the model, i.e., it is related to the variable selection property. Although the choice of these tuning parameters is highly relevant (see, for example Mayr et al., 2012), in this paper we do not consider this issue.

The popularity of the boosting methods has been favored by the availability of user-friendly software. The R (R Development Core Team, 2014) package *mboost* (Hothorn et al., 2010), in particular, provides some routines which allow the users to apply the boosting to several statistical problems (for a complete overview, see Bühlmann & Hothorn, 2007; Hofner et al., 2014). Among these, the routine *glmboost* with argument *family=CoxPH()* tackles time-to-event data when the proportional hazards assumption holds. Another R package which exploits the boosting method within the Cox regression is *CoxBoost* (Binder, 2013). Despite the common underlying gradient boosting idea, these two packages implement different techniques: the former is an adaption of the model-based boosting (Bühlmann & Yu, 2003), while the latter adopts the likelihood-based boosting approach (Tutz & Binder, 2006).

The goal of this study is to contrast the algorithms implemented in these two R packages. For this reason, we focus on time-to-event data, under the proportional hazard assumptions. Although in the context of survival analysis the possibilities offered by these methods exceed the limitations imposed by the Cox model, we note that a large part of the biomedical applications, especially those involving high-dimensional data, relies on this model. The case of high-dimensional data is highly relevant for our study because, in this situation, the traditional statistical tools stop to work and the boosting is a really attractive alternative. For this reason, and because it is the boosting version implemented in *mboost* and *CoxBoost*, in the paper we will pay particular attention to the component-wise boosting.

An important part of our comparison concerns the treatment of mandatory variables, i.e. those variables that for some reasons must be treated in a special way in the statistical analysis. Typical examples are clinical variables in biomedical studies also involving omics data: several papers (see, e.g., Binder & Schumacher, 2008; Boulesteix & Sauerbrei, 2011; De Bin et al., 2014b) show that by considering the clinical variables as mandatory it is possible to obtain better prediction models than by simply merging them with the omics data. We will see that the two R packages implement two different strategies to deal with this issue, and, as a novel contribution of the paper, we will explore the possibility of extending solutions currently only implemented in one package to the other.

The paper is organized as follows: in Section 2 we introduce the Cox model and we briefly review the two boosting algorithms implemented in *mboost* and *CoxBoost*. In Section 3 we contrast these two algorithms showing their similarities and differences as particular cases of the general gradient descending boosting algorithm (Friedman, 2001). The comparison continues in Section 4, where we focus on the issue of mandatory variables and we show how to increase the potential of one boosting package by implementing the solution adopted by the other. A small example with simulated data and two real data applications are shown in Section 5. Some final considerations are reported in Section 6.

2 Methods

2.1 Background

Let us consider the time-to-event data (t, X, δ), where t is the n-dimensional vector of the observed survival times, X the $n \times p$ matrix of the data and δ an n-dimensional vector reporting whether the i-th observed survival time $t^{(i)}$ is censored ($\delta^{(i)} = 1$) or not ($\delta^{(i)} = 0$), $i = 1, \ldots, n$. Hereafter, we suppose, without loss of generalization, that the variables are standardized, i.e. $E[X_j] = 0$ and $Var[X_j] = 1$, $\forall j = 1, \ldots, p$.

To cope with this kind of data, one usually use the Cox model (Cox, 1972) to describe the hazard function $\lambda(t|X)$,

$$\lambda(t|X) = \lambda_0(t) \exp(X^\top \beta), \tag{1}$$

where $\lambda_0(t)$ is the baseline hazard function and β the p-dimensional vector of the regression coefficients. A nice property of the Cox model is that it is not necessary to consider $\lambda_0(t)$ to estimate β, as β is estimated by maximizing the partial log-likelihood

$$pl(\beta) = \sum_{i=1}^{n} \delta_i (X^{(i)})^\top \beta - \log \left(\sum_{l \in R^{(i)}} \exp\{(X^{(l)})^\top \beta\} \right).$$

Here $R^{(i)}$ is the set of the observations at risk at time $t^{(i)}$, while $X^{(i)}$ denotes the i-th observation.

From (1) we note that the hazard function depends in a multiplicative way on β, i.e. the hazard ratio between two observations is proportional over the time. Despite this pretty stringent assumption (usually called proportional hazards assumption), the Cox model is by far the most used tool in the biomedical practice, thanks to the relative ease of interpretation of its regression coefficients. In particular, many approaches related to high-dimensional problems are based on it (Binder & Schumacher, 2008).

The Cox model is also the basis of the two boosting algorithms implemented in the R packages *mboost* and *CoxBoost*. Before analyzing these two specific implementations, we first review the underlying boosting idea, using the concept of functional gradient descending technique (Friedman, 2001). This statistical interpretation of boosting is not unanimous accepted (see, for example, Mease & Wyner, 2008), but it provides a good understanding of the method. Let us call $L(y, F(X))$ a generic loss function, where $F(X)$ is a statistical model. The goal is to estimate $F(X)$ by iteratively updating its value through a base learner $h(y, X)$. The boosting algorithm can be described as follows:

1. initialize the estimate, e.g., $\hat{F}(X) = \text{constant}$;
2. compute the negative gradient vector, $u = -\left.\frac{\partial L(y,F(X))}{\partial F(X)}\right|_{F(X)=\hat{F}(X)}$;
3. compute the update by:
    3.1 fit the base learner to the negative gradient vector, $\hat{h}(u, X)$;
    3.2 penalize the value, $\hat{f}(X) = \nu\hat{h}(u, X)$;
4. update the estimate, $\hat{F}(X) = \hat{F}(X) + \hat{f}(X)$.

In the algorithm, steps from 2 to 4 are repeated m_{stop} times, where m_{stop} denotes the number of boosting iterations. The penalty ν is the other tuning parameter and can assume values between 0 and 1.

This algorithm is very general and can be adapted to several statistical problems. In this paper we focus on applications that are related to the Cox regression and that potentially deal with high-dimensional data. Therefore, in the following we mainly consider the component-wise version of the boosting, in which the algorithm described above is modified in order to update $\hat{F}(X)$ using only one dimension of X in each boosting iteration. In particular, the steps 3 and 4 are applied separately to the different columns of X, generating p possible updates $\hat{f}_j(X)$. An additional step is then implemented to identify which of the p possible updates should be used in the last updating step. Since we restrict our analysis in the Cox regression case, hereafter we consider the parametric version of the boosting, where $F(X)$ is a parametrized class of functions, $F(X, \beta)$ and therefore the update process involves only the estimate of the parameter. The parametric component-wise boosting algorithm is:

1. initialize the estimate, e.g., $\hat{\beta} = (0, \dots, 0)$;
2. compute the negative gradient vector, $u = -\left.\frac{\partial L(y,F(X,\beta))}{\partial F(X,\beta)}\right|_{\beta=\hat{\beta}}$;
3. compute the possible updates by:
    3.1 fit the base learner to the negative gradient vector, $\hat{h}(u, X_j)$;
    3.2 penalize the value, $\hat{b}_j = \nu\hat{h}(u, X_j)$;
4. select the best update j^* (usually that minimizing the loss function);
5. update the estimate, $\hat{\beta}_{j^*} = \hat{\beta}_{j^*} + \hat{b}_j$.

The steps to be repeated m_{stop} times are those between 2 and 5. The quantity $\hat{b}_j$ is also called weak estimator.

2.2 *mboost* for Cox regression

The R package *mboost* is a general tool to implement the boosting. In particular, its function *glmboost* allows to implement the model-based boosting under different models, by implementing the appropriate function for the option *family*. To perform the Cox regression, the pre-built function *CoxPH* is available. Its implementation is based on the work of Ridgeway (1999), who derived the formula for the gradient vector u. The routine *glmboost* is a direct implementation of the functional descending gradient algorithm, in which $L(y, F(X))$ is substituted with the negative partial log-likelihood and $\hat{h}(u, X_j)$ with the least square estimator $(X^\top X)^{-1}X^\top u$. In details, for the Cox regression, the model-based boosting algorithms can be described as follows:

1. initialize $\hat{\beta} = (0, \ldots, 0)$;
2. compute the negative gradient vector, $u_i = \delta_i - \sum_{l \in R^{(i)}} \delta_l \frac{\exp\{X_j^{(l)}\hat{\beta}_j\}}{\sum_{k \in R^{(l)}} \exp\{X_j^{(k)}\hat{\beta}_j\}}$;
3. compute the possible updates by applying the least square estimator to the negative gradient vector, $\hat{b}_j = (X_j^\top X_j)^{-1}X_j^\top u$;
4. select the best update, $j^* = \text{argmin}_j \sum_{i=1}^n (u^{(i)} - X_j^{(i)}\hat{b}_j)^2$;
5. update the estimate, $\hat{\beta}_{j^*} = \hat{\beta}_{j^*} + \nu\hat{b}_{j^*}$.

Steps from 2 to 5 are repeated m_{stop} times.

In practice, starting from $\hat{\beta} = (0, \ldots, 0)$, the algorithm computes the gradient vector of $pl(\beta)$ with respect to $F(X_j, \beta)$, i.e. the direction in which the slope of the partial log-likelihood is locally (in $\hat{\beta}$) more steep (Ridgeway, 1999). A multivariate linear regression is then performed to regress this vector (u) on each X_j. The value of $\hat{b}_j$ which minimizes the residual sum of squares, shrunk by ν, is then used to update $\hat{\beta}$. Roughly speaking, the boosting algorithm “climbs” the partial log-likelihood step by step in the direction which is more correlated with the steepest way to “climb” it. This procedure is iteratively performed m_{stop} times. In case of $p < n$, $\hat{\beta} \to \hat{\beta}_{MPLE}$ as $m_{stop} \to \infty$. The acronym MPLE stays for “maximum partial likelihood estimate”.

2.3 *CoxBoost*: likelihood-based boosting in the Cox model

The R package *CoxBoost* implements in the case of time-to-event data the likelihood-based boosting approach. This approach uses as a loss function the negative L_2-norm penalized partial log-likelihood,

$$pl_{pen}(\beta) = pl(\beta) - 0.5\lambda\beta^\top P\beta,$$

where P is a $p \times p$ matrix usually corresponding to the identity matrix. To save the iterative updates of the parameter estimate, an offset term $\hat{\eta} = X^\top\hat{\beta}$ is added to this log-likelihood, obtaining a function of the form

$$pl_{pen}^{CB}(\beta) = \sum_{i=1}^{n} \delta_i \left[\hat{\eta}^{(i)} + (X^{(i)})^\top\beta - \log\left(\sum_{l \in R^{(i)}} \exp\{\hat{\eta}^{(l)} + X^{(l)}{}^\top\beta\} \right) \right] - \frac{\lambda}{2}\beta^\top P\beta. \tag{2}$$

At each boosting iteration, the maximizer of this function is applied to compute the possible update(s). To better understand the procedure, let us first consider a non-component-wise version, applicable only if $p < n$. The maximizer of (2) is a "weak estimator" because it provides an estimate that highly shrink toward 0 the MPLE. Let focus on the parameter space: as a first iteration, the partial log-likelihood is "shifted" toward the origin by applying the penalty term, obtaining the penalized partial log-likelihood $pl_{pen}(\beta)$. The values of the coordinates of the maximum of $pl_{pen}(\beta)$ are a (small) fraction of related coordinate of the MPLE. The amount of this shrinkage for the different coordinates depends on λ. In particular, a large value of λ has the effect of strongly moving the partial log-likelihood close to the origin, obtaining a penalized partial log-likelihood whose maximum has coordinates which are a very small proportion of those of the MPLE.

The new $\hat{\beta}$ is added to the offset term, and the procedure is re-implemented from this point of the parameter space. Through λ, the partial log-likelihood is now "shifted" toward $\hat{\beta}$ and a new value of the update is computed, moving $\hat{\beta}$ toward the MPLE. Also in this case $\hat{\beta} \to \hat{\beta}_{MPLE}$ as $m_{stop} \to \infty$. It is worth noting that, since λ is constant, the updates to $\hat{\beta}$ become smaller and smaller as far as we proceed with the boosting iterations. Therefore $\hat{\beta}$ continuously approaches the MPLE without reaching it.

The component-wise version follows a similar idea, but the procedure is applied on the p restricted partial log-likelihoods $pl(\beta_j)$. At each boosting iteration, the restricted partial log-likelihoods are "shifted" toward $\hat{\beta}_j$, obtaining the restricted penalized partial log-likelihoods $pl_{pen}^{CB}(\beta_j|\hat{\beta})$. The arguments of the maximums of these functions are the candidate updates, and the one which maximizes the penalized partial log-likelihoods is added to the offset term. In scheme:

1. initialize $\hat{\beta} = (0, \ldots, 0)$;

2-3. compute the possible updates by a first order approximation around 0 of the restricted MPLE $\hat{b}_j^{LB} = \frac{pl_{\beta_j}^{LB}(0|\hat{\beta}_j)}{-pl_{\beta_j\beta_j}^{LB}(0|\hat{\beta}_j)}$;

4. select the best update $j^* = \text{argmin}_{1 \leq j \leq p} pl_{\beta_j}^{LB}(0|\hat{\beta})^2/[-pl_{\beta_j\beta_j}^{LB}(0|\hat{\beta})]$;

5. update the estimate, $\hat{\beta}_{j*} = \hat{\beta}_{j^*} + \hat{b}_{j^*}^{LB}$.

Steps 2-3 – 4 are repeated m_{stop} times. Here $pl_{\beta_j}^{LB}(\beta_j) = \frac{\partial pl_{pen}^{LB}(\beta_j)}{\partial \beta_j}$ denotes the score and $pl_{\beta_j\beta_j}^{LB}(\beta_j) = \frac{\partial^2 pl_{pen}^{LB}(\beta_j)}{\partial \beta_j^2}$ the observed information. The equation at step 4 is the first order approximation around 0 of $pl_{pen}^{LB}(\beta)$, implemented in CoxBoost for computational reasons Binder & Schumacher (2008).

3 Comparison

At a first sight, the two boosting procedures seem quite different. The updates computed within the model-based boosting are based on the correlation between the observations and the negative gradient vector, while for the likelihood-based boosting this procedure involves the maximization of a log-likelihood. Moreover, the penalty term is applied in two completely different ways, multiplied to the updates in the former case, directly on the partial log-likelihood in the latter. In general, these aspects would make the two procedures non-comparable: for example, the penalty parameters would shrink the estimates obtained in each boosting iteration in a very different way depending on the correlation structure of X: we will discuss this aspect in Section 6. Involving only one dimension of X at each iteration, however, the component-wise versions of the boosting procedures implemented in *mboost* and *CoxBoost* are not related to this issue. Moreover, we will see that the form of the linear predictor of the Cox model makes the two boosting procedures even more similar.

As a first step of the comparison, we rewrite the likelihood-based boosting procedure as a functional gradient descending technique:

1. initialize $\hat{\beta} = (0, \dots, 0)$;

2. compute the negative gradient vector $u = \left.\frac{\partial pl(F(X_j,\beta_j))}{\partial F(X_j,\beta_j)}\right|_{\beta=\hat{\beta}}$

3. compute the possible updates as $\hat{b}_j^{LB} = \left(u \left.\frac{\partial F(X_j,\beta_j)}{\partial \beta_j}\right|_{\hat{\beta}_j=0} \right) / \left(- \left.\frac{\partial u \frac{\partial F(X_j,\beta_j)}{\partial \beta_j}}{\partial \beta_j}\right|_{\hat{\beta}_j=0} + \lambda \right)$

4. select the best update $j^* = \text{argmin}_{1\leq j\leq p} \left(u \left.\frac{\partial F(X_j,\beta_j)}{\partial \beta_j}\right|_{\hat{\beta}_j=0} \right)^2 \Big/ \left(- \left.\frac{\partial u \frac{\partial F(X_j,\beta_j)}{\partial \beta_j}}{\partial \beta_j}\right|_{\hat{\beta}_j=0} + \lambda \right)$;

5. update the estimate, $\hat{\beta}_{j*} = \hat{\beta}_{j^*} + \hat{b}_{j^*}^{LB}$.

As before, steps from 2 to 5 are repeated m_{stop} times.

This formulation makes clear that both procedures rely on the negative gradient vector to identify the best "direction" in which the estimate can be improved, and both add this improvement, suitably penalized, to the current value of the estimate. The likelihood-based boosting, in particular, uses the negative gradient vector to derive the score function and the observed information. We saw that these quantities are then used to compute the first order approximation of the maximum penalized partial likelihood estimate around 0. Let us focus on this quantity and contrast it with the update derived with the model-based boosting. The formulas are

$$\hat{b}_j^{CB} = \frac{u \left.\frac{\partial F(X_j,\beta_j)}{\partial \beta_j}\right|_{\hat{\beta}_j=0}}{- \left.\frac{\partial u \frac{\partial F(X_j,\beta_j)}{\partial \beta_j}}{\partial \beta_j}\right|_{\hat{\beta}_j=0} + \lambda} \qquad \text{and} \qquad \nu \hat{b}_j^{MB} = \nu \frac{u X_j}{X_j^\top X_j},$$

respectively. Ignore for the moment the penalty parameters. In the case of linear $F(X_j, \beta_j)$, as for the Cox regression, $\partial F(X_j, \beta_j)/\partial \beta_j = X_j$, and therefore the two numerators are equal. The same is not true for the denominators: the observed information for the Cox model, including the offset term $\hat{\eta} = X^\top \hat{\beta}$, indeed, is

$$- \left.\frac{\partial u \frac{\partial F(X_j,\beta_j)}{\partial \beta_j}}{\partial \beta_j}\right|_{\hat{\beta}_j=0} =$$

$$= \sum_{i=1}^n \delta_i \left\{ \frac{\sum_{l\in R_i} (X_j^{(l)})^2 e^{X^{(l)}\hat{\beta}} \sum_{l\in R_i} e^{X^{(l)}\hat{\beta}}}{\left(\sum_{l\in R_i} e^{X^{(l)}\hat{\beta}}\right)^2} - \frac{\left(\sum_{l\in R_i} X_j^{(l)} e^{X^{(l)}\hat{\beta}}\right)^2}{\left(\sum_{l\in R_i} e^{X^{(l)}\hat{\beta}}\right)^2} \right\},$$

clearly different from the simple X_j^2 of $\hat{b}_j^{MB}$. Using the negative gradient vector in this term as well, indeed, the likelihood-based boosting weak estimator takes into account the concavity of the loss function in the current point of the parametric space $(\hat{\beta})$, while that the model-based boosting estimators uses a sort of parabolic approximation.

Remark 1. It is easy to see that in case of linear regression the two boosting techniques provide the same results. In the linear regression case, i.e., with the Gaussian density as the loss function, $\frac{\partial^2 pl(\beta_j)}{\partial \beta_j^2} = X_j^\top X_j$ and the two denominators are also equal. In case of generalized linear model, instead, the weak estimator for the likelihood-based boosting has form

$$\hat{b}_j^{CB} = \frac{X_j u}{V(\mu) X_j^2},$$

where $\mu = g^{-1}(F(X, \beta))$, with g the canonical link function, and $V(\mu)$ is the variance function (for more details, see McCullagh & Nelder, 1989, Section 2). Also in this case the denominator depends on $\hat{\beta}$ and the two algorithms provide different results. It is possible to obtain with the model-based boosting the same possible updates of the likelihood-based boosting by using a weighted least square estimator instead of the simple least square estimator.

Remark 2. In the previous remark we claimed that the two procedures in the linear regression case provide the same results. This is true for suitable values of the penalty parameters ν and λ. With standardized X, simple algebra shows that the two estimators are equal if $\lambda = n(1 - \nu)/\nu$. In the likelihood-based boosting, λ can assume values from 0, no penalty, to infinity, while in the model-based boosting, ν assumes values between 0 and 1, where 0 corresponds to $\lambda = \infty$ and 1 to $\lambda = 0$. The recommendations of setting λ "sufficiently large" (Binder & Schumacher, 2008) and ν "sufficiently small" (Bühlmann & Hothorn, 2007), therefore, coincide.

Obviously, one could modify the values of the penalty parameters to force the two boosting procedures to give the same results in the Cox regression case as well. This can be done by setting

$$\lambda = \frac{X_j^\top X_j + \nu pl_{\beta_j \beta_j}(0|\hat{\beta})}{\nu}. \tag{3}$$

Since this equation depends on $\hat{\beta}$, it is clear that different values for the penalty parameters should be provided in each boosting iteration. Alternatively, the matrix P should include suitable weights.

Remark 3. The learning path of the two boosting procedures may also differ due to the different choice of which dimension should be updated at each boosting step. In *glmboost* the choice is based on the residuals of the regression of u on X_j, while *CoxBoost* select the dimension which results in the largest decrease of the penalized partial log-likelihood function. As noted before, using a penalized version of the loss function in this step may be advantageous (Mayr et al., 2014).

4 Allowing for mandatory variables

4.1 Background

In the last years, the importance of combining clinical and molecular data in a prediction model becomes clear in the biomedical field, and recent studies contrasted different solutions and methods to profitably use both kinds of data in the model building process (Boulesteix & Sauerbrei, 2011; De Bin et al., 2014b; Truntzer et al., 2014). The main issue related to the combination of clinical and molecular information is the different nature of the data, which belong to the low- and the high-dimensional world, respectively. The consequence is that, if not adequately treated, the risk of "losing" the clinical information among the high number of molecular variables is high (Binder & Schumacher, 2008; Boulesteix & Sauerbrei, 2011).

Both the R packages under investigation tackle this issue by considering the clinical variables as mandatory: in *CoxBoost* there is the possibility, through the argument *unpen.index*, to exclude some variables from the penalization (Binder & Schumacher, 2008). With *mboost*, instead, it is possible to perform a two-step procedure in which the mandatory variables are summarized in a score (in our case, typically the linear predictor of a Cox model) that is later used as an offset in the boosting procedure (Boulesteix & Hothorn, 2010).

In Boulesteix & Sauerbrei (2011), these two strategies are called "favoring" and "residuals", respectively (De Bin et al. (2014b) use the more opportune term "clinical offset" for the latter strategy). These two strategies have some theoretical differences which may influence the model building process and may result more adequate in specific situations. In particular, the "clinical offset" strategy implemented in *mboost* may lead to better results when there is a strong consensus on the effect (regression coefficients) of the clinical (mandatory) variables: the clinical regression coefficients are not modified by the boosting procedure, which only exploits the molecular data to explain the part of the outcome variability not already explained by the clinical model.

The "favoring" strategy implemented in *CoxBoost* follows a different idea, allowing the coefficients of the clinical variables to change during the boosting procedure: at each iteration, the coefficients of the clinical variables are adapted to take into consideration the information provided by the molecular variables. In this way, it may be possible to better integrate the clinical and the molecular information. Binder & Schumacher (2008) defined two ways to perform the stepwise update of the mandatory variables in the likelihood-based component-wise bootstrap framework: in the former, the

$p_0 < n$ mandatory variables are considered in turn with one of the other $p - p0$ variables, with the matrix P associated with the penalty term containing 0 in correspondence to the mandatory variables. The latter way, implemented in *CoxBoost*, instead, consists in updating the regression coefficients of the mandatory variables as a further step before each boosting iteration.

As we stated above, the two strategies to deal with mandatory variables have advantages that may depend on the structure of the data. Therefore, it would be valuable to have the chance to apply both the strategies in both the likelihood- and model-based boosting approaches, to extend the possibilities offered by the two packages. In the following, we consider without loss of generality that the first p_0 columns of X contain the mandatory variables.

4.2 Favoring strategy in *glmboost*

We can allow the regression coefficients of the mandatory variables to vary through the iterations following the model-based boosting. At each boosting iteration, we simultaneously estimate the coefficients of the mandatory and one of the non-mandatory variables, i.e., for each $j = p_0 + 1, \dots, p$, we regress the negative gradient vector on $X_j^+ = (X_1, \dots, X_{p_0}, X_j)$. The choice of the best update is performed as in the regular algorithm, while the penalization is applied only to the last component of the update, that corresponding to the non-mandatory variable:

1. initialize $\hat{\beta} = (0, 0, \dots, 0)$;
2. compute the negative gradient vector u;
3. for each optional variable, compute the possible updates of the coefficients estimates together with the mandatory variables, $\hat{b}_j = (X_j^{+\top} X_j^+)^{-1} X_j^{+\top} u$;
4. select the update which minimizes the residual sum of squares;
5. update the estimate $\hat{\beta}_{[1,\dots,p_0,j^*]} = \hat{\beta}_{[1,\dots,p_0,j^*]} + (\hat{b}_{j*[1]}, \dots, \hat{b}_{j*[p_0]}, \nu \hat{b}_{j*[p_0+1]})$.

Step 2 – 5 are repeated m_{stop} times. The matrix $(X_j^{+\top} X_j^+)^{-1} X_j^{+\top}$ is common in each boosting iteration, and therefore it is sufficient to compute it only once for the $(p - p_0)$ non-mandatory variables. It is worth noting that in our implementation the regression coefficient of the mandatory variables are not shrunken toward 0. We could do that by applying a penalty ν_{clin} to the relative components of the update.

Please note that within this approach, in step 4 we do not select the most significant variable in absolute, but that which explain better the response in concert with the mandatory variables. From a prediction point of view, it means that we select the predictor with largest added predictive value (Boulesteix & Sauerbrei, 2011; De Bin et al., 2014a) instead of that with larger predictive value.

4.3 Clinical offset strategy in *CoxBoost*

To implement the clinical offset strategy into the *CoxBoost* routine we basically need a preliminary step in which we fit a Cox model including the mandatory variables. The linear predictor is then included before the first boosting iteration in $\hat{\eta}$, and the likelihood-based procedure works as usual. Since no iteration will involve the mandatory variables, their regression coefficients are not modified by the boosting procedure. It is worth noting that in this way the boosting works using a penalized restricted partial log-likelihood as a loss function, where the parameters related to the mandatory variables are substituted with their maximum partial likelihood estimates.

1. compute the maximum partial likelihood estimate for the coefficient related to the mandatory variables, $\hat{\beta}_0, \dots, \hat{\beta}_{p0}$;
2. initialize $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_{p_0}, 0, \dots, 0)$;
3. compute, for $j = p_0 + 1, \dots, p$, the potential updates using $\hat{b}_j^{LB}$;
4. determine which $\hat{b}_j^{LB}$ maximizes the penalized partial log-likelihood;
5. update the parameter estimate $\hat{\beta}_j = \hat{\beta}_j + \hat{b}_j^{LB}$ using the $\hat{b}_j^{LB}$ selected in step 4.

Repeat steps 3 – 5 m_{stop} times. Please note that, from a prediction point of view, following this approach we also select the predictor with largest added predictive value.

5 Examples

5.1 Simulated data.

In order to to illustrate the similarities and differences between the likelihood-based and the model-based boosting outlined in Section 3, we conduct a very simple simulation study. We focus on the two-variable regression case, in which it is possible to draw the likelihood function and show the boosting learning path of the component-wise boosting for the linear and the Cox regressions. We generate $n = 200$ observations (X_1, X_2) from a bivariate Gaussian distribution with 0 mean and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

For the linear regression case, we generate the response $y_1, \dots, y_n$ from a Gaussian distribution with variance 1 and mean $\beta_1 X_1 + \beta_2 X_2$, where $\beta_1 = 2$, $\beta_2 = 3$. We center y around its mean and standardize X_1 and X_2. We use the recommended value of 0.1 for the model-based boosting penalty parameter ν and the corresponding $\lambda = (n-1)(1-0.1)/0.1 = 1791$ for the likelihood-based boosting. Please note that the $(n-1)$ substitute n due to the standardization performed with an unbiased estimator of the variance.

For the Cox regression case, instead, we generate the survival times t through the formula

$$t = -0.1 \frac{\log(a)}{e^{\beta_1 X_1 + \beta_2 X_2}},$$

where a is generated from a uniform distribution between 0 and 1. If t is smaller than a random draw from an exponential distribution with rate 0.1, it is the observed time, while if it is larger, the observation is considered censored and the value generated from the exponential distribution is the observed time. For more details on this simulation model, see Binder & Schumacher (2008). In this case, we deliberately choose a non-optimal value for the penalty parameter in order to show more efficaciously the different learning path. In fact, we set $\nu = 0.25$ for the model-based boosting and the corresponding $\lambda = (n-1)(1-0.25)/0.25 = 597$ for the likelihood-based boosting.

Figure 1 shows that, in case of linear regression, the component-wise versions of the model-based and the likelihood-based boosting procedures provide the same results. Conversely, in Figure 2, we clearly see that the equivalence does not hold in the case of Cox regression. Please note that the magnitude of the difference depends on the value of the parameters ν and λ. Reducing ν and, in parallel, increasing λ, indeed, lead to learning paths always more similar, with both approaching the learning path of the least angle regression (Efron et al., 2004) for $\nu \to 0$ and $\lambda \to \infty$, respectively.

Table 1 reports the values obtained for $\hat{\beta}$ in the first 10 boosting iterations. Starting from $\hat{\beta} = (0, 0)$, we obtain the update candidates $\hat{b}_1^{[1]} = 0.204$, $\hat{b}_2^{[1]} = 0.219$ for *CoxBoost* and $\nu\hat{b}_1^{[1]} = 0.178$, $\nu\hat{b}_2^{[1]} = 0.191$ for *glmboost*. As we have seen, the differences lies in the denominator of the two estimators: the numerators coincide and are equal to 142.004 and 151.811, for $\hat{b}_1^{[1]}$ and $\hat{b}_2^{[1]}$, respectively. Here the superscript $[m]$ denotes the m-th boosting iteration. For *CoxBoost* these values are divided by the information, 694.845 and

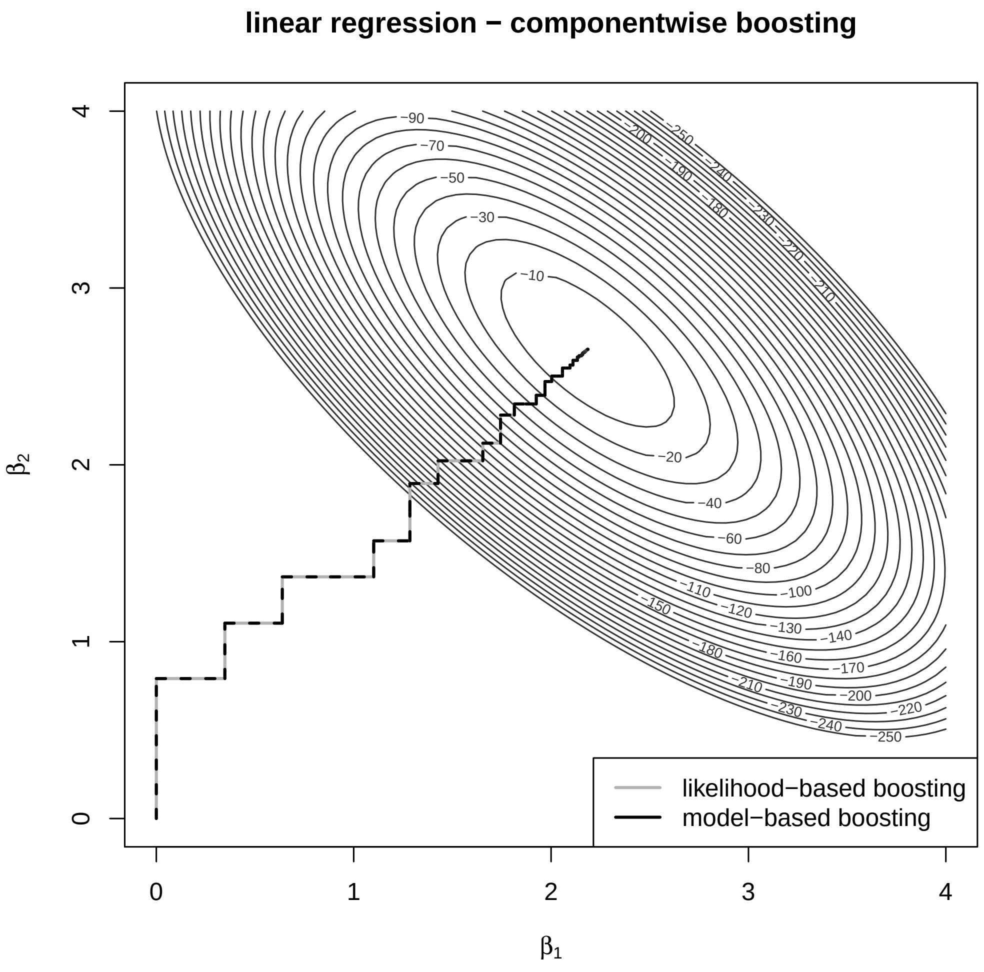

Figure 1: Learning path of the component-wise versions of the two boosting approaches in the linear regression case. The contour lines represent the levels of the normalized log-likelihood.

692.533, respectively, while for *glmboost* they are divided by $X_j^\top X_j/\nu$, which, since we are using standardized predictors, corresponds to $(n-1)/\nu = 796$, despite the index j. The results are different unless equation (3) is satisfied. In particular, keeping λ fixed, this equation requires ν to be equal to 0.286 for $\hat{b}_1^{[1]}$ and to 0.287 for $\hat{b}_2^{[1]}$: in Table 1, 0.219 is indeed $0.191 \times 0.287/0.25$. In the first step, both the techniques select the second candidate, being the values of the loss function -29.021 versus -33.279 in *CoxBoost* (approximation of the profile penalized partial log-likelihood) and 67.442 versus 52.961

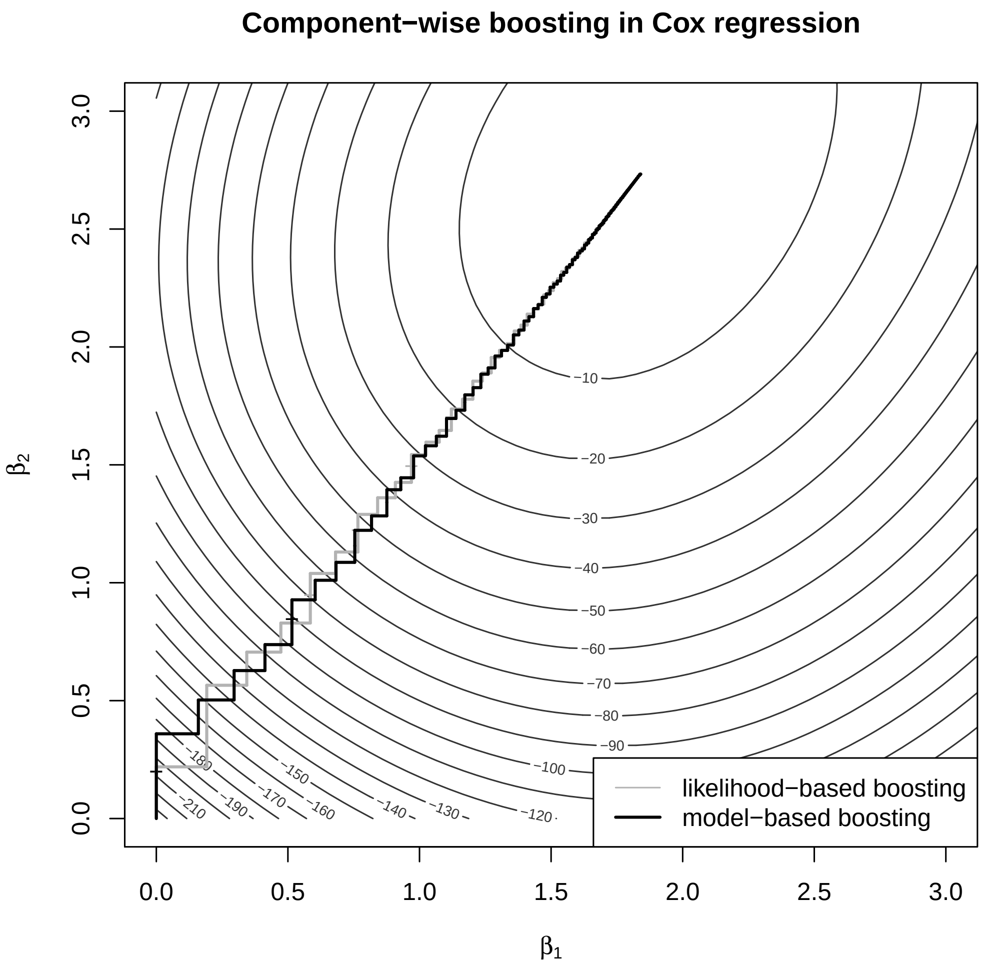

Figure 2: Learning path of the component-wise versions of the boosting approaches in the Cox regression case. The contour lines represent the levels of the normalized partial log-likelihood.

in *glmboost* (residual sum of squares). This is not always the case, as we can see from in the second step: in this case the candidates are $\hat{b}_1^{[2]} = 0.192$, $\hat{b}_2^{[2]} = 0.192$ for *CoxBoost* and $\nu\hat{b}_1^{[2]} = 0.168$, $\nu\hat{b}_2^{[2]} = 0.168$ for *glmboost*, which lead to the values of loss function -25.385 versus -25.348 (*CoxBost*) and 54.349 versus 53.486 (*glmboost*). In this case, the former techniques update $\hat{\beta}_1$, while the latter again $\hat{\beta}_2$.

boosting	*CoxBoost*		*mboost*	
iteration	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_1$	$\hat{\beta}_2$
0	0.000	0.000	0.000	0.000
1	0.000	0.219	0.000	0.191
2	0.192	0.219	0.000	0.359
3	0.192	0.402	0.160	0.359
4	0.192	0.565	0.160	0.503
5	0.344	0.565	0.296	0.503
6	0.344	0.706	0.296	0.627
7	0.473	0.706	0.413	0.627
8	0.473	0.829	0.413	0.738
9	0.585	0.829	0.515	0.738
10	0.585	0.939	0.515	0.836

Table 1: Estimates of the Cox regression coefficients in the first 10 steps of the boosting procedures.

5.2 Colon cancer data

In this example, we would like to show the possible advantages of using a favoring strategy over the default clinical offset in the model-based boosting. We use the data on colon cancer presented in a study by Marisa et al. (2013) and publicly available in the *ArrayExpress* web repository with reference number E-GEOD-39582.

The dataset contains 566 observations split into a training (443) and a test set (123). Out of these observations, 10 were discarded due to missing values, obtaining a training set of 439 and a validation set of 117 observations. In particular, the effective sample size, i.e. the number of observations with an event ($\delta_i = 1$) is 141 and 36. We have information about 4 clinical variables, namely *sex*, *age*, *subtype* and *stage*. The latter two are categorical variables with 6 and 4 modalities, and we need to transform them in dummy variables. The molecular data, instead, consist in 54675 gene expressions determined on Affymetrix U133 Plus 2.0 chips.

In order to implement the model-based boosting, we centered the continuous variables, namely the age and the gene expressions. Please note that for centering the variables we use the means computed on the training set only. We set the penalty term $\nu = 0.1$ and select the number of boosting iterations m_{stop} via a repeated 10-fold cross-validation, which consists of merging the results of several 10-fold cross-validation replications in order to make the results robust with respect to the specific split of the data in the 10 folds (see also Boulesteix et al., 2013).

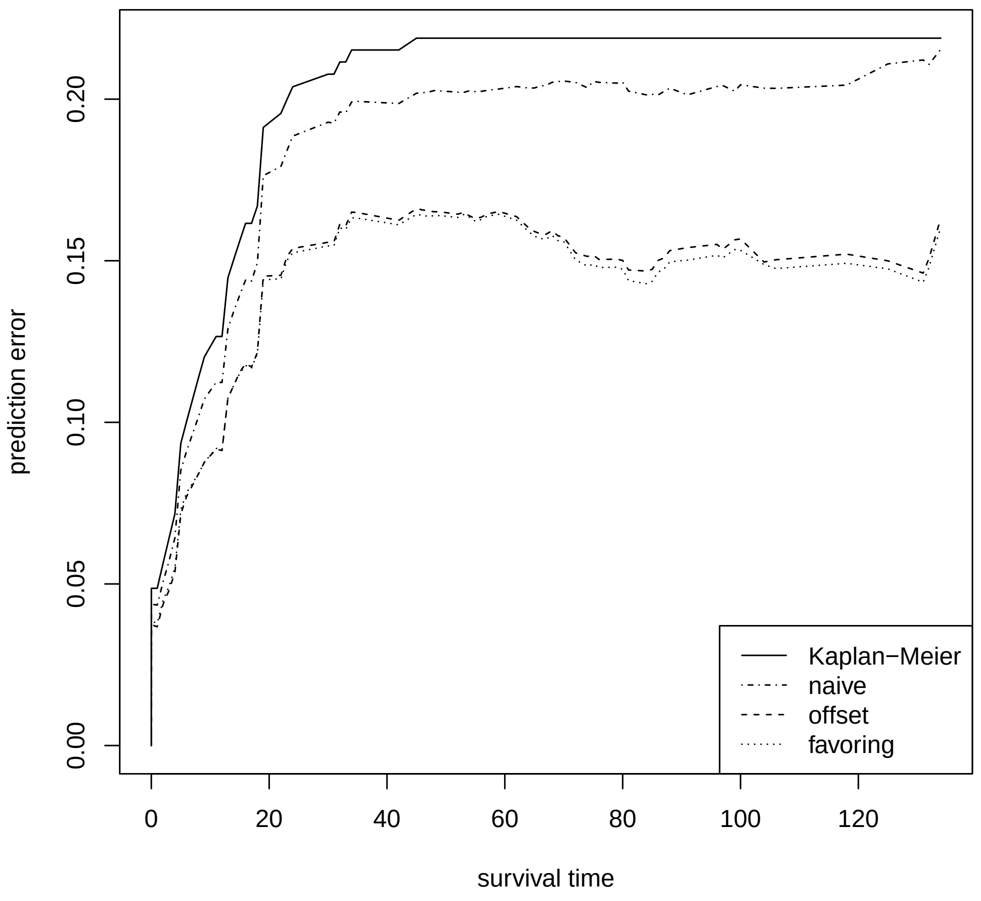

Figure 3: Prediction error curves based on the Brier score computed in the test set for the null model (continuous line) and for the boosting models obtained following a naive (dot-dashed line), a clinical offset (dashed line) and a favoring (dotted line) strategy.

We evaluated the performance of the models obtained by following different strategies to combine clinical and molecular data in terms of Brier score (Graf et al., 1999), a time-dependent quadratic score for survival data. The results are summarized in the so called "prediction error curves" using the R packages *pec* (Mogensen et al., 2012). It is worth noting that all models are trained in the training set and their performance evaluated in the test set only. As we can see in Figure 3, in this example there is a small improvement in performing the favoring strategy (dotted line) rather than following the clinical offset strategy (dashed line). This example, moreover, shows very clearly the importance of treating the clinical and the molecular variables in a different way in the model building process: implementing a naive strategy in which the two kinds of data are simply merged together ignoring their differences, indeed, we obtain a definitely worse prediction (dot-dashed line).

5.3 Advanced prostate cancer

In this example we see a case in which the likelihood-base boosting benefits from the implementation of the clinical offset strategy with respect to the favoring one. In this case, we focus on a low dimensional example, in particular on a well-known dataset about advanced prostate cancer first analyzed using a Cox model by Byar & Green (1980). The data are publicly available at *http://portal.uni-freiburg.de/imbi/Royston-Sauerbrei-book/*, as supplementary file of Royston & Sauerbrei (2008). In this study, information on 12 variables were collected, some without any technical measurement (namely *age* and *history of cardiovascular disease*, continuous and binary, respectively) and other with the help of some instruments (*weight*, *systolic blood pressure*, *diastolic blood pressure*, *size of the primary tumor*, *serum acid phosphatase*, *hemoglobin*, *Gleason stage-grade category* as continuous variables, *performance status*, *presence of bones metastases* and *tumor stage* as binary variables). We consider the former two (*age* and *history of cardiovascular disease*) as mandatory variables, the latter as optional. It is clear that this split is quite artificial, but we are mainly interested in the effect of the boosting procedure rather than in the substantive research question.

The dataset contains information about 475 patients with 338 events. Since a separation between training and test sets is not available, we randomly split the observations with a 2/3 ratio, obtaining a training set with 317 observations and a test sets with sample size equal to 158. To reduce the dependence on the specific split, we repeat the analysis 100 times, generating different training and test sets. For each of these split, we standardized all the continuous variables and codified the binary variables using the values $(-1; 1)$. The observations in the test sets are standardized using the standard deviation computed on the corresponding training set.

For each split, we compute the value of the penalty parameter λ using the routine *optimCoxBoostPenalty* of the package *CoxBoost*. We compute this value only once, and then use it for all the boosting implementations of the different strategies. The effect of the parameter λ on the results, indeed, is not particularly relevant, and Binder & Schumacher (2008) claimed that it is only important that it has the right magnitude. For the computation of the number of boosting iterations m_{stop}, instead, we use a 10-fold cross-validation procedure, specific for each strategy implementation. Again, the models are trained using the training set and evaluated using the test set only in terms of Brier score.

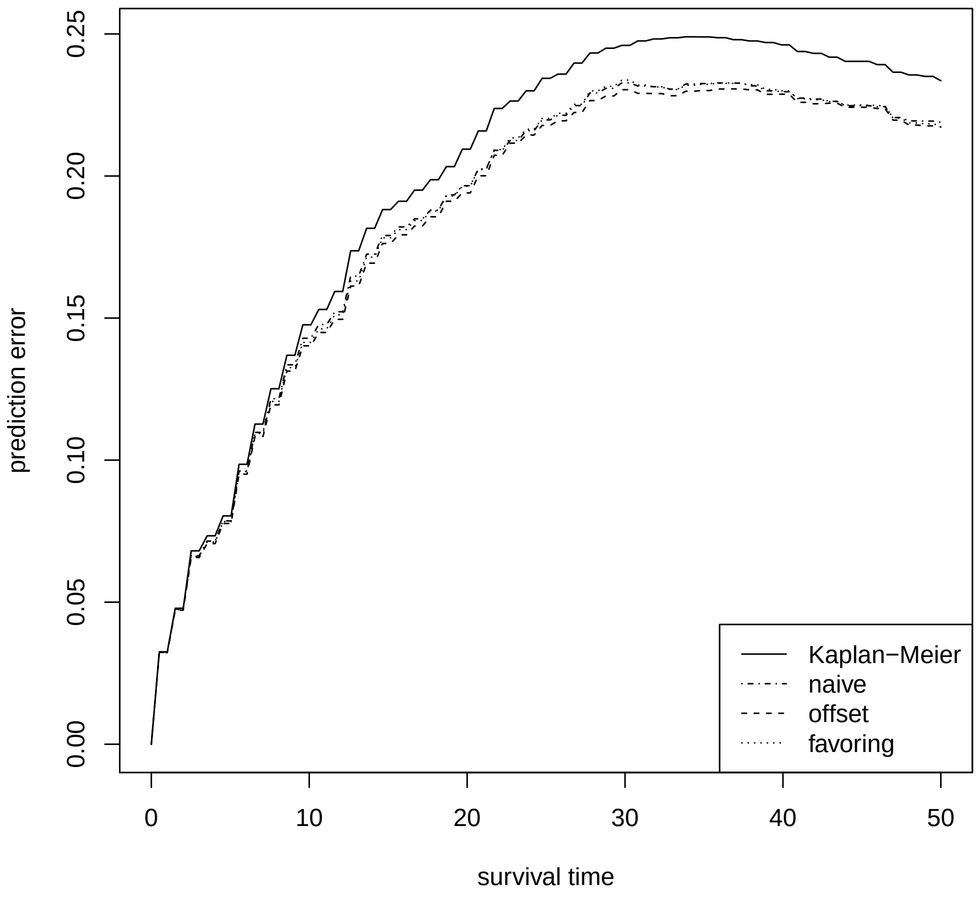

Figure 4: Prediction error curves obtained by averaging the Brier scores computed in the several test sets for the null model (continuous line) and for the boosting models obtained following a naive (dot-dashed line), a clinical offset (dashed line) and a favoring (dotted line) strategy.

Figure 4 shows the results. In this case, the clinical offset strategy (dashed line) performs slightly better than the favoring one (dotted line). It is worth noting that, in this example, both the mandatory and the optional parts are low-dimensional, and therefore there is no strong advantage is using a special strategy to combine them. In particular, as expected, the favoring and the naive strategy lead to the same prediction error.

6 Discussion

In this paper, we contrasted the model- and likelihood-based boosting algorithms for the Cox regression implemented in the R packages *mboost* and *CoxBoost*. We showed that they follow different learning paths, conversely to the linear regression case in which the same result is produced, providing $\lambda = n(1-\nu)/\nu$. It is worth noting that the equivalence in the linear regression case only works for the component-wise version of the boosting. In the non-component-wise version, when we update all the dimensions of β simultaneously, indeed, the two weak estimators have the form

$$\hat{b}^{CB} = (X^\top X + \lambda P)^{-1} X^\top u \qquad \text{and} \qquad \nu \hat{b}^{MB} = \nu (X^\top X)^{-1} X^\top u,$$

for the likelihood- and the model-based boosting, respectively. While the model-based penalty ν affects all the dimensions in the same way, the penalty λ penalizes the dimensions with respect to the correlation structure of X. Consider $P = I_p$, the identity matrix used as default in *CoxBoost*. The weak estimator $\hat{b}^{CB}$ is then a ridge estimator: in the regression procedure, when the response is projected on the orthonormal basis of the explanatory variables (columns of X), the penalty term shrinks the coordinates (inversely) proportionally to the variance of the related principal components. This means, in particular, that the term λ penalizes (shrinks) more the coefficients related to principal components with low variance. If we look at the predictive values obtained through the two algorithms, denoting with B the orthonormal basis of the columns of X, we obtain

$$\begin{aligned} \hat{y}^{MB} &= B \, \mathrm{diag}(1 - (1-\nu)^{m+1}) B^\top y \\ \hat{y}^{CB} &= B \, \mathrm{diag}(1 - (1 - \tfrac{d_j}{d_j + \lambda})^{m+1}) B^\top y, \end{aligned}$$

where d_j, $j = 1, \ldots, p$ is the j-th eigenvalue of the matrix $X^\top X$ (divided by n, is the variance of the principal component) and m indicates the number of iterations performed. Substituting m with 0 we obtain the formula for the single step update. It is worth noting that, due to its stage-wise nature, the algorithm of boosting ridge regression leads to a different penalization (and, therefore, to different estimates) than the usual ridge regression (Tutz & Binder, 2006).

We can obtain a uniform penalization with the likelihood-based boosting by setting $P = (1/n)X^\top X$, provided that the columns of X are centered in 0 and standardized. In this case, in particular, $(1/n)X^\top X$ represents the correlation matrix of X.

It is worth noting that it is possible to use a penalized least square estimator within the model-boosting algorithm as well. This solution seems to gain popularity (see, e.g., Hofner et al., 2014), because it allows to better handle categorical variables. Please note that in this case the whole penalization is a combination of the effect of λ and ν. Another issue related to categorical variables is related to their variance. In this paper, we considered standardized X, but we saw in the examples that in practical situations we need a standardize process. The likelihood-based boosting, in particular, needs all X_j having variance 1. For this reason, in the Example 5.2 we codified the dummy variables as $(-1; 1)$. In case of completely balanced observations, the variance of the binary variables is then equal to 1. Unfortunately, this happens rarely in practice. A possible solution which does not require the standardization of X is to substitute P with the covariance matrix of X or, in case of component-wise version, with $\operatorname{diag}((1/n)X^\top X)$. In this way we would standardize the binary variables using their observed standard deviation as well.

We would like to remind that the main focus of this paper was on the Cox regression case, in which the effect of the regression parameter is relatively easy to understand. For these reasons, here we did not consider more sophisticated boosting versions which include non-linear effects (Schmid & Hothorn, 2008; Hofner et al., 2013) or are based on a tree-based approach (Ridgeway, 2010). A different possibility to treat survival data using *glmboost* is provided by Hothorn et al. (2006): instead of using the partial log-likelihood as the loss function, they modify the boosting algorithm for the linear regression (L_2Boosting Bühlmann & Yu, 2003), adapting the loss function (Gaussian log-likelihood) and the least square estimator by adding some weights to take into account the censored nature of the data (Hothorn et al., 2006; Bühlmann & Hothorn, 2007). In particular, the inverse probability of censoring weights (Van der Laan & Robins, 2003) are used. This approach does not necessarily rely on the Cox model (it may do through a specific definition of the weights) and therefore it is not considered here.

We finally note that in this paper we considered the theoretical similarities and differences of the two boosting algorithms, without trying to evaluate their performance in specific cases. The complexity of the reality makes almost impossible to identify all the situations in which one algorithm performs better than the other: from this point of view, it is positive to have different solutions which enrich the practitioner's toolbox. For this reason, a relevant part of this paper is devoted to the extensions of the solutions available in only one R package to the other. Nevertheless, it is important to use these possibilities wisely, without falling into the temptation of trying all of them and then reporting only the results for the method which more supports our theory.

Aknowledgements

We are grateful to Anne-Laure Boulesteix for the valuable advice and suggestions.

References

Binder, H. (2013). *CoxBoost: Cox models by likelihood based boosting for a single survival endpoint or competing risks.* R package version 1.4.

Binder, H. & Schumacher, M. (2008). Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models. *BMC Bioinformatics* **9**, 14.

Boulesteix, A.-L. & Hothorn, T. (2010). Testing the additional predictive value of high-dimensional molecular data. *BMC Bioinformatics* **11**, 78.

Boulesteix, A.-L., Richter, A. & Bernau, C. (2013). Complexity selection with cross-validation for lasso and sparse partial least squares using high-dimensional data. In *Algorithms from and for Nature and Life.* Springer, pp. 261–268.

Boulesteix, A. L. & Sauerbrei, W. (2011). Added predictive value of high-throughput molecular data to clinical data and its validation. *Briefings in Bioinformatics* **12**, 215–229.

Breiman, L. (1998). Arcing classifier. *The Annals of Statistics* **26**, 801–849.

Bühlmann, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics* **34**, 559–583.

Bühlmann, P. & Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting. *Statistical Science* **22**, 477–505.

Bühlmann, P. & Yu, B. (2003). Boosting with the L_2 loss: regression and classification. *Journal of the American Statistical Association* **98**, 324–339.

Byar, D. & Green, S. (1980). The choice of treatment for cancer patients based on covariate information. *Bulletin du Cancer* **67**, 477–490.

Cox, D. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)* **34**, 187–220.

De Bin, R., Herold, T. & Boulesteix, A.-L. (2014a). Added predictive value of omics data: specific issues related to validation illustrated by two case studies. *BMC Medical Research Methodology* **14**, 117.

De Bin, R., Sauerbrei, W. & Boulesteix, A. L. (2014b). Investigating the prediction ability of survival models based on both clinical and omics data: two case studies. *Statistics in Medicine* **33**, 5310–5329.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics* **32**, 407–499.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation* **121**, 256–285.

Freund, Y. & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc.

Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics* **28**, 337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* **29**, 1189–1232.

Graf, E., Schmoor, C., Sauerbrei, W. & Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine* **18**, 2529–2545.

Hofner, B., Hothorn, T. & Kneib, T. (2013). Variable selection and model choice in structured survival models. *Computational Statistics* **28**, 1079–1101.

Hofner, B., Mayr, A., Robinzonov, N. & Schmid, M. (2014). Model-based boosting in R: a hands-on tutorial using the R package mboost. *Computational Statistics* **29**, 3–35.

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A. & Van Der Laan, M. J. (2006). Survival ensembles. *Biostatistics* **7**, 355–373.

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M. & Hofner, B. (2010). Model-based boosting 2.0. *The Journal of Machine Learning Research* **11**, 2109–2113.

Marisa, L., de Reyniès, A., Duval, A., Selves, J., Gaub, M. P., Vescovo, L., Etienne-Grimaldi, M.-C., Schiappa, R., Guenot, D., Ayadi, M., Kirzin, S., Chazal, M., Fljou, J.-F., Benchimol, D., Berger, A., Lagarde, A., Pencreach, E., Piard, F., Elias, D., Parc, Y., Olschwang, S., Milano, G., Laurent-Puig, P. & Boige, V. (2013). Gene expression classification of colon cancer into molecular subtypes: characterization, validation, and prognostic value. *PLoS Medicine* **10**, e1001453.

Mayr, A., Binder, H., Gefeller, O. & Schmid, M. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine* **53**, 419–427.

Mayr, A., Hofner, B. & Schmid, M. (2012). The importance of knowing when to stop. A sequential stopping rule for component-wise gradient boosting. *Methods of Information in Medicine* **51**, 178–186.

McCullagh, P. & Nelder, J. (1989). *General Linear Models.* Chapman and Halls, London.

Mease, D. & Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *The Journal of Machine Learning Research* **9**, 131–156.

Mogensen, U. B., Ishwaran, H. & Gerds, T. A. (2012). Evaluating random forests for survival analysis using prediction error curves. *Journal of Statistical Software* **50**, 1–23.

R Development Core Team (2014). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Ridgeway, G. (1999). *Generalization of boosting algorithms and applications of Bayesian inference for massive datasets.* Ph.D. thesis, University of Washington.

Ridgeway, G. (2010). *gbm: generalized boosted regression models.* R package version 1.6.

Royston, P. & Sauerbrei, W. (2008). *Multivariable Model-building: a pragmatic approach to regression anaylsis based on fractional polynomials for modelling continuous variables.* Wiley, Chichester.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning* **5**, 197–227.

Schmid, M. & Hothorn, T. (2008). Flexible boosting of accelerated failure time models. *BMC Bioinformatics* **9**, 269.

Truntzer, C., Mostacci, E., Jeannin, A., Petit, J.-M., Ducoroy, P. & Cardot, H. (2014). Comparison of classification methods that combine clinical data and high-dimensional mass spectrometry data. *BMC Bioinformatics* **15**, 385.

Tutz, G. & Binder, H. (2006). Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics* **62**, 961–971.

Van der Laan, M. J. & Robins, J. M. (2003). *Unified Methods for Censored Longitudinal Data and Causality*. Springer, New York.