Machine Learning Approaches to Survival Analysis: Case Studies in Microarray for Breast Cancer

Liu Yang and Kristiaan Pelckmans, *Member, IACSIT*

***Abstract*—Cox Proportional Hazard model and the associated Partial Likelihood criterion have been the main tool of inference for data of survival studies. There are many powerful techniques in modern machine learning which could be applied on the survival studies and increase the performance. This paper consider the problem of comparing different techniques for inference with large dimensional data of breast cancer. Besides an overview of the different techniques, numerical experiments are presented. Liu Yang managed to implement the algorithms and the experiments. Kristiaan Pelckmans managed to complete the theoretical part.**

***Index Terms*—Machine learning, survival analysis, boost-ing, SVM.**

I. Introduction

The analysis of data observed in survival studies has been an important topic in applied sciences, as well in theoretical work on probabilistics and stochastics. The work of B. Cox and co-workers in the 70s has been instrumental [1], [2]. A useful survey of work in that general area has been the topic of numerous textbooks, amongst which [3] relative to the work in statistical inference, methods of Machine Learning (ML) have been studied rather scarcely in this setting. However, ideas as regularisation, boosting and sparsity promoting priors have found its way into this field. Historically seen, developments can be divided into three loosely related directions:

- (Penalized Likelihood) The most prevalent approach takes the traditional method of choice for semiparametric inference of survival analysis, and endows this approach with a mechanism penalising unlikely deemed solutions.
- (Boosting) Boosting approaches have found their way from purely machine learning algorithms into the realm of statistical inference [4]. It comes hence as no surprise that such approaches have been examined in the context of survival analysis. Broadly speaking, a boosting approach mixes simple solutions ('weak learners') into one global, strong model. This mixing is regulated by an iterative approach, zooming in on the more intricate parts of the model whenever needed.
- (SVMs) A third line of research is based on using the techniques behind Support Vector Machines (SVMs) to tackle inference in survival models. This line of research was investigated in some details in [5], [6]. The former approach relates the task to ranking approaches, via the so-called model of *transformation models*. The three corner stones of SVMs are (i) the use of techniques of convex optimisation and duality; (ii) the use of an objective function which approaches the risk of prediction (prognosis) directly; (iii) the use of nonlinear kernels for reduction to a linear, high-dimensional problem.

This paper aims to identify the more powerful approaches when confronted with (i) high-dimensional data (covariates), (ii) survival data originating from breast cancer studies. Hereto, we present mostly empirical results obtained on a number of publicly available datasets. One crucial element of our study is how we score a certain method. That is, which criterion for selecting an appropriate method is used? Broadly speaking, there are two goals one can aim for:

1) *Prognosis:* Predict the distribution of events of a fresh subject.
2) *Recovery*: Which are the factors regulating the risk of the observed phenomenon.

ML approaches do basically aim for *prognosis*, while classical approaches typically aim for the latter.

In this paper, the following notational conventions are used: random variables are denoted as capital letters $X, Y, Z, \ldots$. Vectors are denoted in boldface $x, y, \ldots$. Deterministic quantities are represented as lowercase letters $i, n, f, \ldots$.

A. Basic Setup

The data is represented as a set of size n of tuples.

$$\{(\mathbf{x}_i, Y_i, \delta_i)\}_{i=1}^n, \tag{1}$$

where

- The i th subject is represented by a vector of covariates $\mathbf{x}_i \in \mathbb{R}^p$, where $p \geq 1$ is its dimension. In the studied cases, p is typically larger than n.
- The i th subject experienced an event at a time $T_i \geq 0$. However, this time is only observed if the subject stayed in the study for long enough. Otherwise it is censored.
- The time of censoring is $C_i \geq 0$.
- One observes only T_i if its uncensored, or $Y_i = \min(T_i, C_i)$.
- The variable $\delta_i = \{0,1\}$ indicates if the observation is censored ($\delta_i = 0$) or not ($\delta_i = 1$). That is $\delta_i = I(T_i > C_i)$ where I denotes the identicator function, that is $I(z) = 1$ if z holds true, and equals zero otherwise.

In this paper, only right censoring is dealt with. Moreover, the presented ML techniques assume independence of the n

Manuscript received April 10, 2014; revised July 14, 2014. This work was supported in part by Swedish Research Council under contract 621-2007-6364.

Liu Yang is with the Department of Information Technology, Uppsala Universtiy, 75105 Uppsala, Sweden (e-mail: sjtuly@gmail.com).

Kristiaan Pelckmans is with the Department of Information Technology, Uppsala Universtiy, 75105 Uppsala, Sweden.

observations, which means that censoring is either deterministic or independent of the other $n-1$ subjects.

The rest of this paper is organized as follows. Section II introduces the Cox Proportional Hazard Model and its basic properties. Three penalized approaches are surveyed in this section. Section III reviews a number boosting methods for survival analysis. Section IV briefly describes the transformation-model based MINLIP algorithms, extending the Support Vector Machine (SVM) to the analysis of survival data. Section V introduces a screening method called Sure Independence Screening (SIS). Section VI gives details of our experiments as well as the empirical results.

II. Inference in the Proportional Hazard Model

The basic model of interest in the analysis of survival data is the Proportional Hazard (PH) model [2]. Let $S(t) = P(T > t)$ be the survival function [2]. It is defined as a probability on the domain $t \in [0, \infty)$, it has range $S(t) \in [0,1]$. The hazard function [2] $h(t)$ is then defined as:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \le T \le t + \Delta t \mid T \ge t)}{\Delta t}. \quad (2)$$

Then $S(t)$ as a function of $h(t)$ is:

$$S(t) = P(T > t) = \exp\left\{-\int_0^t h(\tau) d\tau\right\}, 0 < t < \infty. \quad (3)$$

When T is continuous and the derivative S exists, we have a density function $f(t) = \frac{dS(t)}{dt}$ and hence $S(t) = \int_t^\infty f(x)dx$. The cumulative hazard function [2] is $\Lambda(t) = \int_0^t h(\tau) d\tau$. A PH model has the form [2]:

$$h(t, \mathbf{x}) = h_0(t) \exp\left(\boldsymbol{\beta}^T \mathbf{x}\right), \quad (4)$$

where $h_0(t)$ is the baseline hazard function which depends on time but not on the covariates. The exponential term depends on the covariates but not on time. In this model, when we are interested in the effects of the covariates on survival, we do not need to specify the form of the baseline function. The Cox PH model is called a semi-parametric model, as some assumptions are made on the parametric influence of the covariates, but as no form is pre-specified for the baseline hazard. Taking logs, an additive form is obtained [2]: $\log h(t, \mathbf{x}) = \log h_0(t) + \boldsymbol{\beta}^T \mathbf{x}$. This shows that one unit's change of a covariate will have equal effect independent of the values of the other covariates. If a model does not satisfy this assumption, there may be interaction effects in the covariates. Checking for interaction terms becomes cry cumbersome when confronted with high-dimensional data.

Parameter estimates in the PH model are obtained by maximizing Cox's Partial Likelihood (PL) [1]:

$$L_n(\beta) = \prod_{i:\delta_i = 0} \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{\sum_{Y_j \ge Y_i} e^{\boldsymbol{\beta}^T \mathbf{x}_j}}. \quad (5)$$

Note that this function is convex in β.

A. L_1 Penalised Partial Likelihood

The L_1 penalized method, (also referred to as LASSO [7] [8], can also be added to the PL function of Cox PH model [9], giving

$$L_1(\boldsymbol{\beta}) = \log L_n(\boldsymbol{\beta}) - \lambda_1 \| \boldsymbol{\beta} \|_1. \quad (6)$$

So that estimates β are given by maximising this function. For the study on Cox PH model, we will use 10-fold cross-validated partial likelihood [10] to tune this λ_1.After fitting the model, the non-zero regression coefficients are retained. In a second step, only those covariates are used to fit the final model.

B. L_2 Penalised Partial Likelihood

Instead of adding a L_1 penalized term, ridge regression uses a L_2 term.

$$L_2(\boldsymbol{\beta}) = \log L_n(\boldsymbol{\beta}) - \lambda_2 \| \boldsymbol{\beta} \|_2. \quad (7)$$

The penalized L_2 method will generally not perform as well as LASSO [8] when only a few of the presented high-dimensional covariates are significantly non-zero [9]. The method used for tuning λ_2 is similar to L_1 section.

C. Other Penalisation Schemes

There are three requirements for an ideal penalty function: unbiasedness, sparsity and continuity [11].

The smoothly clipped absolute deviation (SCAD) penalty can satisfy all of them. It is defined by its derivative by [12]:

$$p'_\lambda(\beta_i) = \left(I\left(|\beta_i| \le \lambda\right) + \frac{\left(a\lambda - |\beta_i|\right)}{(a-1)\lambda} I\left(|\beta_i| > \lambda\right) \right), a > 2. \quad (8)$$

An explicit form of the SCAD penalty is given by [13]:

$$p_\lambda(\beta_i) = \begin{cases} \lambda |\beta_i| & \text{if } |\beta_i| \le \lambda, \\ \lambda \dfrac{\left(a - |\beta_i| / 2\lambda\right)}{a-1} & \text{if} \lambda < |\beta_i| \le a\lambda, . \\ \lambda \dfrac{a^2 \lambda}{(a-1) 2|\beta_i|} & \text{if} a\lambda < |\beta_i|, \end{cases} \quad (9)$$

In this equation, both λ and a need to be tuned. However, a good default value $a = 3.7$ is given by [11]. The penalised PL will then be:

$$L_{SCAD}(\boldsymbol{\beta}) = \log L_n(\boldsymbol{\beta}) - \sum_{i=1}^{d} p_\lambda(\beta_i). \quad (10)$$

Maximizing this function the results in our third estimator.

III. Boosting for Survival Analysis

Boosting is a technique of Machine Learning (ML). There are two basic elements:

Weak Learner.

Mixing and Reweighting.

The surprising feat is that such approach might avoid *overfitting* altogether.

A. Weak Learners

Weak learners have the form $f_i : \mathbb{R}^d \to \mathbb{R}$. The utility function $f^n : \mathbb{R}^d \to \mathbb{R}$ is often used in boosting algorithms. The utility function is a weighted sum of the weak rankings $f^n = \sum_{i=1}^{n} \alpha_i f_i$, where α_i are the parameters for the corresponding weak learners. By using the utility function we can define different kinds of error. Boosting algorithms try to find the weak learners which lead to smallest error. The weak learners are not always a high accuracy learner, but must be simple and better than a completely random method. Some examples are given as following:

Simple 0-1 weak learner: $f_i(\mathbf{x}) = I(\mathbf{x} \in S)$, where S is a set which satisfies some conditions.

Logistic regression: $f_i(\mathbf{x}) = \dfrac{1}{1+\exp\left(-\beta^T \mathbf{x}\right)}$, where β is found by logistic regression of the weak learning problem.

Regression trees: $f_i(\mathbf{x}) = \lambda_i sign\left(\mathbf{x}_{k_i} - \tau_i\right)$. A decision tree is composed of a stump function $f_i(\mathbf{x})$ at every non-leaf node i. For each stump i, it is parametrized with a parameter λ_i, and a threshold τ_i and a feature index k_i.

Each stump will output a decision and move to the next stump, until the leaf nodes.

B. Boosting as Functional Gradient Decent

Gradient boosting techniques [14] are usually used for fitting high-dimensional models. The intuition is to see boosting as a form of steepest descent for minimizing a loss function. The predictor function is found by minimizing the value of a specified loss function L over the training set. The $\mathcal{L}(f)$ [14] indicates the error by the predictor f. The negative partial log-likelihood is normally chosen as the loss function for Cox PH model. Instead of starting with a weighted sum of weak learners, this approach models the boosting process by the following recursion. Define g^m as the negative gradient of f^{m-1} for the $m-th$ step, $g^m = \nabla\left(f^{m-1}\right)$, then $f^m = f^{m-1} + \alpha_m g^m$, where $\alpha_m = \arg\min_\alpha \mathcal{L}(f^{m-1} + \alpha g^m)$. And $f_0 = 0$. In each step, f^m is constructed by following the negative gradient negative gradient and with optimal step size α_m. Since one does not have a closed form solution of g^m, we train a learner h^m which approximates the negative gradient g^m. Define $\tilde{\alpha}^m$ as $\tilde{\alpha}^m = \arg\min_\alpha \mathcal{L}(f^{m-1} + \alpha h^m)$, and update $\tilde{f}^m = \tilde{f}^{m-1} + \tilde{\alpha}^m h^m$, where $\tilde{f}^0 = 0$. In general, this method will use all covariates.

An alternative approach of gradient boosting is componentwise boosting [14]. It uses a linear predictor $f(\mathbf{x}, \beta) = \beta^T \mathbf{x}$.

In each boosting step, only one element of β is updated. The one to be updated is chosen by evaluating fits to the gradient, the resulting fits will indicate an element which improves the overall fit the most. And this also lead to the sparseness of the solutions [15], since many coefficients will be estimated to zeros.

C. CoxBoosting

The aim of the CoxBoost [16] is to estimate the parameter vector $\boldsymbol{\beta}$ for the linear predictor $F(\mathbf{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \mathbf{x}$. In the boosting step k, there are q_k predetermined candidates sets of covariates with indices $I_{kl} \subseteq \{1, ..., p\}, l = 1, ..., q_k$. For each of the q_k sets there will be an update of the parameters for the corresponding covariates. The one set which improves the fit the most will be chosen as the final update. Let $\boldsymbol{\beta}_{k-1} = (\hat{\beta}_{k1}, ..., \hat{\beta}_{kp})'$ be the actual estimate of the overall parameter vector β after step k-1 of boosting, and $\hat{\eta}_{i,k-1} = \mathbf{x}_i' \boldsymbol{\beta}_{k-1}$ be the corresponding linear predictors. The potential updates $\boldsymbol{\gamma}_{kl}$ for the elements of $\boldsymbol{\beta}_{k-1}$ are obtained by maximizing the penalized log-partial-likelihood $l_{pen}(\gamma_{kl})$ [17]. The penalty matrices P_{kl} [16] can be specified separately for each boosting step and each candidate set. The formulas of this method are given in [16], [17].

D. RankBoost

RankBoost is an algorithm which combines weak rankings of the instances into a single highly accurate ranking [18]. On each interation a procedure named weak learn is called in order to produce a weak ranking f_i.

The number of instances is denoted as n. We assume that d ranking features are given, each feature defines a linear ordering of the patients. RankBoost computes in each iteration t a set of weight $\{b_{i,j}\}_{i,j}$ which describes the importance of the comparable pairs for $i = 1...n; j = 1...n$. A comparable pair means that we can measure the order between these two instances in the pair. The weight of a non-comparable pair will be set to zero. For survival analysis, if a patient $i = 1...n; j = 1...n$ has event before $\mathbf{x}_j$, we call $(\mathbf{x}_i, \mathbf{x}_j)$ a comparable pair. The utility function is used to order the survival time. $f(\mathbf{x}_0) > f(\mathbf{x}_1)$ means that $\mathbf{x}_0$ is expected to have longer survival time than $\mathbf{x}_1$.The utility function f is a weighted sum of the weak rankings as we defined in the weak learner section. The algorithm attempt to find the weights with a small quantity of wrongly ordered pairs, called the ranking loss and denotes $rloss(f)$. The sample ranking loss is defined formally in [18]: And we choose the weak rankings $h_t(\mathbf{x}) = I(f_{k(t)}(\mathbf{x}) > \theta)$. Here $k(t)$ indicates the feature which would lead to the smallest loss.

IV. Inference for Transformation Models

A transformation model relates a function of the covariates to the response variable through a monotone increasing mapping. Inference then concerns recovery of both this function, as well as this mapping. This makes this setting different tom methods of GLM, where the monotone mapping (or the so called *link* function) is given in advance.

Let $l:\mathbb{R}\to\mathbb{R}$ be a strictly increasing function with Lipschitz constant $\ell<\infty$, and let $u:\mathbb{R}^{\mathrm{d}}\to\mathbb{R}$ be a function of the covariates $\mathbf{x}\in\mathbb{R}^d$. Let ϵ be a random variable noise. A noisy transformation model has the form:

$$Y=l(u(\mathbf{x})+\epsilon). \tag{11}$$

The problem is to estimate a utility function u and a transformation function l from a set of observations $\{(\mathbf{x}_i,Y_i)\}_{i=1}^n$. To solve this problem, we can learn a transformation model with minimal Lipschitz constant of l. When we define $u(\mathbf{x})=\mathbf{w}^T\mathbf{x}$, this takes equivalently forms. See Formula (12) in section 3.3 of [5]. We will refer to this model as MINLIP. Then we also can have some extensions to MINLIP. To obtain the sparseness of representations in the coefficients, an extension called MINLIP $_L1$ is to use L_1 penalty [Tibshirani1996], see Formula (33) in section 3.5 of [6]. An alternative to it, is incorporating positivity constraints of the parameters, see Formula (34) in section 3.5 of [6]. We call this one MINLIP $_p$. Then we consider to relate the MINLIP model to the SVM method [19]. Based on the rankSVM model for ranking or preference learning [20], and a similar ranking SVM model for survival problem [21], we can have a MINLIP&SVM mixed model, we note it as MODEL 1, see Formula (20) in section 3.1 of [22]. By including regression constraints as in [23] and [24], MODEL 1 can be modified as MODEL 2, see Formula (23) in section 3.2 of [22]. Now considering high dimensional data, to avoid overfitting the model, feature selection is included in MODEL 2 by constraining the weights w to positive weights and will be denoted as MODEL 2P. See Formula (20) in section 3.1 of [22].

V. Sure Independence Screening

A. SIS

A nature idea is for high-dimensional modeling is dimensionality reduction, or feature selection. Let $\mathbf{m}=(\mathbf{m}_1,...\mathbf{m}_p)^T$ be a p-vector obtained by the componentwise regression $\mathbf{m}=\sum_{i=1}^n\mathbf{x}_i^TY_i$. For any given $\gamma\in(0,1)$, $d=[\gamma n]$, where $[\gamma n]$ denotes the integer part of γn. We define a submodel [25]:

$$M_\gamma=\{1\le i\le p:|\mathbf{m}_i|\text{is among the first }[\gamma\text{n}]\text{largest}\}. \tag{12}$$

This method can shrink the full model with size $p>>n$ down to a submodel M_γ with size $d<n$. We call it Sure Independence Screening (SIS) [25]. In practice we may choose d to be conservative like $n-1$ or $n/\log n$. Sometimes we could also break the limit, set $\gamma>1$ and $d>n$ to drive the model work better. There are two steps, first we use SIS method to reduce dimensionality from p to d, which is usually below the sample size n. Then we use the methods such as LASSO and SCAD to train a Cox model.

B. ISIS

An extension of SIS is an iterative SIS (ISIS) [25]. It is designed to overcome some weak points of SIS, such as missing some important features. The ISIS work as the following steps: In the first step, we select a subset A_1 of k_1 features, using SIS-SCAD or SIS-LASSO methods. By using the regression coefficients on A_1, we can calculate a new response Y_1.

In the next step, we apply the methods again as in the first step, to Y_1 and the remaining $p-k_1$ features. And this result in a subset A_2 and a new Y_2.

We can keep doing this until we get L disjoint subsets. And we compute $A=\bigcup_{i=1}^L A_i$. In practice we can choose the largest L such as $|A|<n$.

VI. Empirical Validation

A. Performance Measures

A Concordance Index is the numerical measure used to score the fitted model. It is the probability of concordance between the predicted and the observed survival. It is defined as:

$$C_n(f)=\frac{\sum_{i:\delta_i=0}\sum_{Y_j>Y_i}I(f(\mathbf{x}_i)<f(\mathbf{x}_j))}{|\varepsilon|}. \tag{13}$$

Here $|\varepsilon|$ denotes the number of the pairs which have $Y_i<Y_j$ when Y_i is not censored. The indicator function $I(\pi)=1$ if π holds, and 0 otherwise. $f(\mathbf{x}_i)$ is the predicted survival time for patient *i*. The concordance probability (CP) [26], [27] for the Cox PH model is defined as:

$$CP_n(\beta)=\log\prod_{T_j>T_i}\frac{\exp(\boldsymbol{\beta}^T\mathbf{x}_i)}{\exp(\boldsymbol{\beta}^T\mathbf{x}_i)+\exp(\boldsymbol{\beta}^T\mathbf{x}_j)}. \tag{14}$$

The log-rank statistic tests for equality of survival of two groups G_1 and G_2. It can be also used to score how good the unity function f can separate the high risk from the low risk subjects as follows. Let $\bar{f}$ be the median value of the vector $\{f(\mathbf{x}_i)\}_{i=1}^n$. Suppose that we consider the following two groups: $G_{1,f}=\{i:f(\mathbf{x}_i)\ge\bar{f}\}$ and $G_{2,f}=\{i:f(\mathbf{x}_i)<\bar{f}\}$. Let $j=1,...J$ be the index of the distinct survival times of events in either group. For each time point j, define:

$$\begin{cases}N_{1,j}(f)=\sum_{i:i\in G_{1,f},\delta_i=0}I(Y_i>Y_j)\\ N_{2,j}(f)=\sum_{i:i\in G_{2,f},\delta_i=0}I(Y_i>Y_j)\\ O_{1,j}(f)=\sum_{i:i\in G_{1,f},\delta_i=0}I(Y_i=Y_j)\\ O_{2,j}(f)=\sum_{i:i\in G_{2,f},\delta_i=0}I(Y_i=Y_j).\end{cases} \tag{15}$$

Let $N_j(f) = N_{1,j}(f) + N_{2,j}(f)$ and $O_j(f) = O_{1,j}(f) + O_{2,j}(f)$ .Then the log-rank statistic is given as:

$$logrank(f) = \frac{\left(\sum_{j=1}^{J} \left(O_{1,j} - \frac{O_j}{N_j} N_{1,j} \right) \right)^2}{\sum_{j=1}^{J} \frac{O_j N_{1,j} N_{2,j} (N_j - O_j)}{(N_j)^2 (N_j - 1)}} \quad (16)$$

And this statistic follows a χ^2 distribution

B. Tuning

Tuning is an important concern in the application of this technique. But it also introduces a problem: if a method has many variables to tune, the result might hinge on the method used for tuning rather than on the technique used for inference. In order to circumvent this issue, we allow such method to be tuned to a single tuning parameter. In case of penalised Proportion Hazard Methods and SVM-based methods, a natural choice concerns the choice of the regularization parameter $\lambda > 0$ [28]. In case of boosting, one optimises for the number of iterations. The remaining tuning parameters (if any) are set to a reasonable default value. Assessment of the performance follows the following strategy. The data is repeatedly and randomly split in training and test data. The training data is used for training the model, as well for tuning the appropriate parameter using cross-validation. The final model is then computed on this training set, making use of this optimally tuned hyper-parameters. Once the final model is obtained, its accuracy is scored based on the corresponding test-set. This procedure is randomised m times (where we set $m = 20$), and the median and variance of the m scores is given in the table.

C. Artificial Data

We generate some artificial survival data with specified proportion of the informative covariates. The data includes a training set of 100 patients and a test set of another 100 patients. We set the number of covariates to 100, and assign different specified number k of the informative covariates. We generate the covariates from the standard normal distribution, $\mathbf{x}_i \backsim N(0,1)$.

And we can calculate the real time T_i for patient i by:

$$T_i = \frac{-\log \delta}{10 \exp\left(x_i^1 + x_i^2 + ..., x_i^k \right)} \quad (17)$$

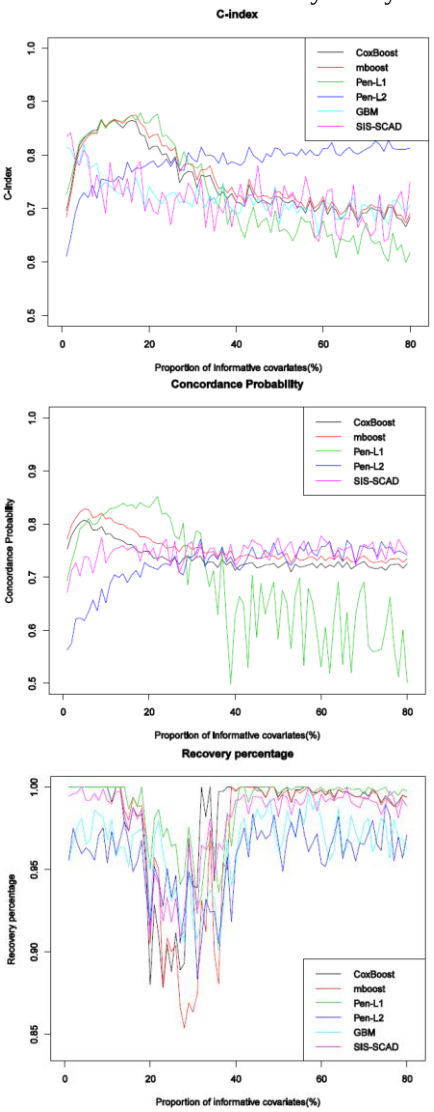
where δ is a random value generated from 0-1 uniform distribution, x_a^b is the $b-th$ covariate for patient a . k is the specified number of the informative covariates. The censoring time is randomly generated from the exponential distribution with rate $1/10$ .Then we use the right censoring rule, comparing the real times and censoring times, to calculate the final survival times and the censoring indicators. By applying the methods on the artificial data, we study how the performances change as the proportion of the informative covariates increases. Table I shows the procedure.

TABLE I: TEST ON PROPORTION OF THE INFORMATIVE COVARIATES

Tests by 5 methods (CoxBoost, mboost, gbm, Pen-L1 and Pen-L2)

Given n the size of data set, R range to search
Set the maximum repeat times m (by default m=50)
Set the number of covariates $p = 100$
FOR $(i \in R)$
FOR $k = 1, 2, \ldots, m$
(1) Generate the artificial training set and test set by n , p , i
(2) Apply the 5 methods on the training set
(3) Calculate the predictions on the test set
(4) Compute the measures CI and CP
(5) Find the $top - i$ predictors returned by the methods
(6) Compare them with the real i informative covariates to get the accuracy
END
Compute the median of the measures
END
Plot the Figures

See the result in Fig. 1. There are three figures show how the medians of C_n , CP_n and the recovery percentage change as the proportion of the informative covariates increases. And the last figure shows the error bar of the "best" method Pen-L_1 (LASSO). Based on the figures, LASSO shows good performance on sparse data, while L_2 performs better as the sparseness decreases [10], [9]. And the three boosting methods performs better on the recovery ability.



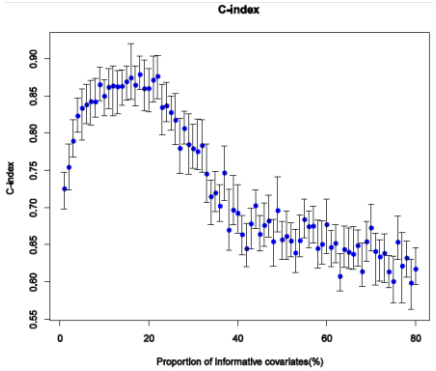


Fig. 1. Experiments on proportion of informative features.

D. Studies of Breast Cancer

Results are computed for the following publicly accessible datasets:

- NSBCD: The Norway/Stanford Breast Cancer Data set is given in [29]. In this database there are survival data of 115 women who have breast cancer, and 549 intrinsic genes introduced in [29] were measured. In the 115 patients, 33% (38) have experienced an event during the study. Missing values were imputed by 10-nearest neighbor method.
- DBCD: The Dutch Breast Cancer Data set is given in [30], and is a subset of the data from [31]. There are survival data of 295 women who have breast cancer. The measures of 4919 gene expression were taken from the fresh-frozen-tissue bank of the Netherlands Cancer Institute. All the ages of the patients are smaller than or equal to 52 years. The diagnosis was made between 1984 and 1995 without previous history of cancer. The median of follow-up time was 6.7 years (range 0.05-18.3). In the 295 patients, 26.78% (79) have experienced an event during the study.
- DLBCL: The diffuse large-B-cell lymphoma data set is from [32]. There are survival data of 240 patients who have diffuse large-B-cell lymphoma. 7399 different gene expression measurements are given. The median of follow-up time was 2.8 years. In the 240 patients, 58\% have experienced an event during the study.
- Veer: The survival data of sporadic lymph-node-negative patients with their gene expression profiles is given in [31]. There are 78 patients with 4751 gene expressions selected from the 25,000 genes on the microarray. 44 patients remained free of disease after their diagnosis for an interval of at least 5 years. The mean of follow-up time for these patients was 8.7 years. 34 patients had developed distant metastases within 5 years, mean time to metastases 2.5 years.
- Vijver: The data set of 295 consecutive patients with primary breast carcinomas is from [31]. All patients had stage I or II breast cancer and were younger than 53 years old. They gave the previously determined 70 marker genes that are associated with the risk of early distant metastases in young patients with lymph-node-negative breast cancer. The median follow-up among all 295 patients was 6.7 years (range, 0.05 to 18.3). There were no missing data. 88 patients have experienced an event during the study.
- Beer: The survival data of 86 patients with primary lung adenocarcinomas is from [33]. There are 7129 expressed genes selected from Affymetrix hu6800 microarrays. 76 patients have experienced an event during the study.
- AML: The survival data of acute myeloid leukemia patients is from [34]. It contains 116 patients with acute myeloid leukemia and 6283 genes. 71 patients have experienced an event during the study.

We use 50 random permutations. For every data set, there are tuples $\{(\mathbf{x}_i, Y_i, \delta_i)\}_{i=1}^n$. $\mathbf{x}_i$ are the covariates, Y_i contains the survival time, δ_i is the censoring indicator. The procedure is described in Table II:

TABLE II: EXPERIMENTS FOR ARTIFICIAL DATA

Procedure of the experiments
Given the dataset include $\{(\mathbf{x}_i, Y_i, \delta_i)\}_{i=1}^n$
FOR $i = 1, 2, \ldots, 50$
(1) Partition the set into 1/3 as test set and 2/3 as training set
(2) Tune the models on training set only, get the optimal parameters
(3) Apply the methods with the optimal parameters on the training set
(4) Get the predictions on the test set based on the learnt model
(5) Calculate the performance measures (C_n, CP_n, logrank)
END
Compute the median values and the standard deviations of the measures

TABLE III: EXPERIMENTS ON THE 7 DATA SETS BY DIFFERENT METHODS

Dataset	Method	C_n	Logrank	CP_n
NSBCD	Minlip	0.7248±0.0231	5.0796	
	Minlip*P*	0.7390±0.0325	5.3807	
	Model2	0.7466±0.0113	6.1193	
	Model2*P*	0.7351±0.0105	4.9463	
	PCR	**0.7739±0.0234**	7.3779	0.7687±0.0098
	SPCR	0.7687±0.0298	7.1358	0.7603±0.0104
	PLS	**0.7823±0.0312**	**7.5111**	**0.7698±0.0187**
	RankBoost	0.7528±0.0299	6.789	
	CoxBoost	0.7242±0.0233	6.0133	0.7087±0.0045
	mboost	0.7045±0.0132	3.4523	0.6456±0.0076
	GBM	0.7381±0.0541	5.3698	0.6483±0.0215
	MinlipL 1	0.5910±0.0135	NaN	
	Lasso	0.6932±0.0569	3.0452	0.6610±0.0481
	PHL2 (Matlab)	0.6957±0.0423	2.5224	0.7011±0.0194
	PHL2 (*R*)	0.6897±0.0465	2.3352	0.5194±0.0034
	SCAD	0.6874±0.0257	3.164	0.6587±0.0133
	ISIS-SCAD	0.6913±0.0387	3.4523	0.6631±0.0125
DBCD	Minlip	0.6823±0.0245	5.4396	
	Minlip*P*	0.7124±0.0243	8.4900	
	Model2	0.7293±0.0342	10.2383	
	Model2*P*	0.6274±0.0287	2.5367	
	PCR	0.7277±0.0192	10.5851	0.7138±0.0183
	SPCR	0.7278±0.0105	11.1827	0.7142±0.0123
	PLS	**0.7379±0.0178**	**13.6349**	0.7197±0.0204

	RankBoost	0.7225±0.0267	9.4563	
	CoxBoost	0.7297±0.0323	11.0423	0.7129±0.0154
	mboost	0.7145±0.0187	8.4564	0.6632±0.0203
	GBM	0.7032±0.0453	8.7558	0.6653±0.0224
	Minlip*L* 1	0.5573±0.0223	1.6569	
	Lasso	**0.7351±0.0359**	12.046	**0.7640±0.0168**
	PHL2 (Matlab)	0.7275±0.0532	10.6103	0.7133±0.0329
	PHL2 (*R*)	0.7233±0.0145	10.9763	0.7034±0.0196
	ISIS-SCAD	0.7173±0.0423	9.2341	0.6932±0.0287
	SCAD	0.7114±0.0543	8.3452	0.6894±0.0154
DLBCL	Minlip	0.5577±0.0102	0.9393	
	Minlip*P*	0.5945±0.0079	3.9829	
	Model2	0.6168±0.0143	4.4273	
	Model2*P*	0.6016±0.0165	2.5715	
	PCR	0.5934±0.0098	NaN	0.5901±0.0105
	SPCR	0.5950±0.0108	NaN	0.5923±0.0056
	PLS	0.5000±0.0000	NaN	0.5000±0.0000
	RankBoost	0.6213±0.0178	3.2576	
	CoxBoost	0.6023±0.0169	3.7532	0.5603±0.0174
	mboost	0.5874±0.0091	3.3453	0.5443±0.0083
	GBM	0.5932±0.0227	3.2764	0.6033±0.0142
	Minlip*L* 1	0.5531±0.0087	1.1712	
	Lasso	0.6034±0.0382	4.2352	**0.7203±0.0144**
	PHL2 (Matlab)	**0.6506±0.0225**	**8.6053**	0.6347±0.0127
	PHL2 (*R*)	0.6397±0.0211	4.5767	0.6118±0.0175
	ISIS-SCAD	**0.6467±0.0714**	7.4322	0.7043±0.0459
	SCAD	0.6133±0.0576	6.4761	0.7016±0.0745
Veer	Minlip	0.5640±0.0187	1.0345	
	Minlip*P*	0.6675±0.0264	2.0579	
	Model2	0.6673±0.0312	**2.5740**	
	Model2*P*	0.6577±0.0365	1.4140	
	PCR	0.6800±0.0542	NaN	0.6698±0.0353
	SPCR	**0.6863±0.0478**	NaN	**0.6734±0.0403**
	PLS	0.5000±0.0000	NaN	0.5000±0.0000
	RankBoost	0.6211±0.0214	2.0194	
	CoxBoost	0.6290±0.0187	1.5732	0.5811±0.0248
	mboost	0.6324±0.0164	2.0432	0.5546±0.0239
	GBM	0.6697±0.0623	2.4387	0.6629±0.0378
	Minlip*L* 1	0.5897±0.0213	0.9409	
	Lasso	0.6032±0.0661	2.4523	0.6325±0.1356
	PHL2 (Matlab)	0.6399±0.0346	1.8182	0.6175±0.0254
	PHL2 (*R*)	0.6273±0.0239	2.1498	0.6147±0.0678
	ISIS-SCAD	**0.6794±0.0423**	2.1237	0.6529±0.0781
	SCAD	0.6312±0.0231	1.7684	0.6349±0.0651

Vijver	Minlip	0.5929±0.0192	1.7353	
	Minlip*P*	0.6074±0.0275	2.8562	
	Model2	0.6185±0.0203	2.1863	
	Model2*P*	0.5894±0.0174	1.8480	
	PCR	0.6247±0.0387	NaN	0.6094±0.0277
	SPCR	0.6260±0.0473	NaN	0.6116±0.0225
	PLS	0.6272±0.0410	NaN	0.6188±0.0364
	RankBoost	0.6097±0.0229	1.9372	
	CoxBoost	0.6125±0.0178	2.1235	0.5874±0.0106
	mboost	0.5874±0.0238	1.8632	0.5423±0.0254
	GBM	0.6314 ± 0.0579	3.3732	0.6113±0.0463
	Minlip*L* 1	0.6286±0.0248	3.1272	
	Lasso	**0.6466±0.0594**	**3.4623**	**0.6562±0.0626**
	PHL2 (Matlab)	0.6300±0.0260	NaN	0.6203±0.0195
	PHL2 (*R*)	0.6277±0.0357	1.2453	0.6158±0.0123
	ISIS-SCAD	0.6232±0.0212	1.1987	0.6066±0.0276
	SCAD	0.6251±0.0336	1.3764	0.6107±0.0379
Beer	Minlip	0.6572±0.0254	1.5439	
	Minlip*P*	0.6790±0.0237	1.4854	
	Model2	0.7282±0.0229	1.6670	
	Model2*P*	0.6860±0.0217	1.0582	
	PCR	0.6771±0.0196	NaN	0.6601±0.0165
	SPCR	0.6644±0.0287	NaN	0.6319±0.0229
	PLS	0.6883±0.0274	NaN	0.6801±0.0253
	RankBoost	0.7135±0.0204	1.5743	
	CoxBoost	0.7214±0.0149	1.8923	0.6985±0.0204
	mboost	0.6823±0.0174	1.7982	0.6415±0.0237
	GBM	0.06931±0.0354	1.9712	0.6832±0.0557
	Minlip*L* 1	0.5643±0.0276	0.6902	
	Lasso	0.6911±0.1776	1.9432	0.6201±0.0743
	PHL2 (Matlab)	**0.7313±0.0296**	**2.3047**	**0.7132±0.0218**
	PHL2 (*R*)	0.7197±0.0876	2.0345	0.6433±0.0521
	ISIS-SCAD	0.7233±0.0423	2.2166	0.6953±0.0276
	SCAD	0.7114±0.0257	1.9875	0.6770±0.0319
AML	Minlip	0.5496±0.0106	0.5175	
	Minlip*P*	0.5533±0.0103	0.4698	
	Model2	0.5649±0.0159	1.6522	
	Model2*P*	0.5385±0.0065	1.5904	
	PCR	0.5501±0.0154	NaN	0.5413±0.0104
	SPCR	0.5609±0.0203	NaN	0.5577±0.0196
	PLS	0.5669±0.0164	NaN	0.5595±0.0109
	RankBoost	0.5265±0.0073	0.6234	
	CoxBoost	0.5387±0.0094	0.9842	0.5412±0.0048
	mboost	0.5514±0.0128	0.4562	0.5232±0.0078
	GBM	**0.6353±0.0533**	**2.5674**	**0.6497±0.0693**

	Minlip*L* 1	0.5053±0.0074	0.4544	
	Lasso	0.6109±0.0577	1.9764	0.6308±0.0230
	PHL2 (Matlab)	0.5374±0.0177	0.8904	0.5211±0.0098
	PHL2 (*R*)	0.5397±0.0046	1.3248	0.5190±0.0105
	ISIS-SCA D SCAD	**0.6311±0.0198**	2.3354	0.5797±0.0432
		0.6215±0.0314	1.9869	0.5991±0.0335

All the results are put in Table III. Based on this table we can compare the performance of these methods for each data set. If we focus on the c-index, LASSO and ISIS-SCAD work best in average. For extremely high dimensional data like DLBCL, Lung, etc. the L_2 penalized Cox PH model often performs better than the others. The transformation model based methods performs better than many classical methods in average. The Boosting algorithms perform between the classical group and the penalized group. For some data sets like AML they can reach outstanding scores. The other two measures (CP_n and Logrank) also support the result by c-index. Note that for some methods the CP_n cannot be calculated.

REFERENCES

[1] D. R. Cox, "Partial likelihood," *Biometrika*, vol. 62, no. 2, pp. 269–276, 1975.

[2] D. R. Cox *et al.*, "Regression models and life tables," *JR stat soc B*, vol. 34, no. 2, pp. 187–220, 1972.

[3] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, 2011.

[4] R. E. Schapire, "The boosting approach to machine learning: An overview," *Nonlinear estimation and classification*, pp. 149–171, 2003.

[5] V. Van Belle, K. Pelckmans, J. A. Suykens, and S. Van Huffel, "Minlip: Efficient learning of transformation models," *ICANN*, pp. 60–69, 2009.

[6] V. Van Belle, K. Pelckmans, J. A. K. Suykens, S. Van Huffel, "Learning transformation models for ranking and survival anal- ysis," *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[7] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[8] R. Tibshirani *et al.*, "The lasso method for variable selection in the cox model," *Statistics in medicine*, vol. 16, no. 4, pp. 385–395, 1997.

[9] J. Goeman, "Penalized: L1 (lasso) and l2 (ridge) penalized estimation in glms and in the cox model," *R package version*, 2008.

[10] J. J. Goeman, "L1 penalized estimation in the cox proportional hazards model," *Biometrical Journal*, vol. 52, no. 1, pp. 70–84, 2010.

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

[12] J. Fan, "Comments on wavelets in statistics: A review by a. antoniadis," *Journal of the Italian Statistical Association*, vol. 6, pp. 131–138, 1997.

[13] B. A. Johnson and L. Peng, "Rank-based variable selection," *Journal of Nonparametric Statistics*, vol. 20, no. 3, pp. 241–252, 2008.

[14] G. Ridgeway, "Generalized boosted regression models," *R package version*, vol. 1, no. 5, p. 7, 2006.

[15] B. Hofner, A. Mayr, N. Robinzonov, and M. Schmid, "Model-based boosting in r: a hands-on tutorial using the r package mboost," *Computational Statistics*, pp. 1–33, 2012.

[16] H. Binder, "Coxboost: Cox models by likelihood based boosting for a single survival endpoint or competing risks," *R package version*, vol. 1, 2009.

[17] H. Binder, A. Benner, L. Bullinger, and M. Schumacher, "Tailor- ing sparse multivariable regression techniques for prognostic single-nucleotide polymorphism signatures," *Statistics in medicine*, vol. 32, no. 10, pp. 1778–1791, 2013.

[18] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *The Journal of machine Learning Research*, vol. 4, pp. 933–969, 2003.

[19] V. N. Vapnik, *Statistical learning theory*, 1998.

[20] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," *Advances in Neural Information Processing Systems*, pp. 115–132, 1999.

[21] V. Van Belle, K. Pelckmans, J. Suykens, and S. Van Huffel, "Support vector machines for survival analysis," *CIMED*, 2007, pp. 1–8.

[22] V. Van Belle, K. Pelckmans, S. Van Huffel, and J. A. Suykens, "Support vector methods for survival analysis: a comparison between ranking and regression approaches," *Artificial intelligence in medicine*, vol. 53, no. 2, pp. 107–118, 2011.

[23] P. K. Shivaswamy, W. Chu, and M. Jansche, "A support vector approach to censored targets," in *Proc. Seventh IEEE International Conference on Data Mining*, 2007, pp. 655–660.

[24] F. M. Khan and V. B. Zubek, "Support vector regression for censored data (SVRc): a novel tool for survival analysis," in *Proc. Eighth IEEE International Conference on Data Mining*, 2008, pp. 863–868.

[25] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911, 2008.

[26] M. Gönen and G. Heller, "Concordance probability and discriminatory power in proportional hazards regression," *Biometrika*, vol. 92, no. 4, pp. 965–970, 2005.

[27] V. C. Raykar, H. Steck, B. Krishnapuram, C. Dehing-Oberije, and P. Lambin, "On ranking in survival analysis: Bounds on the concordance index," *NIPS*, 2007.

[28] A.-L. Boulesteix and T. Hothorn, "Testing the additional predic-tive value of high-dimensional molecular data," *BMC bioinformatics*, vol. 11, no. 1, p. 78, 2010.

[29] T. Sørlie, R. Tibshirani, J. Parker *et al.*, "Repeated observa- tion of breast tumor subtypes in independent gene expression data sets," *Proceedings of the National Academy of Sciences*, vol. 100, no. 14, pp. 8418–8423, 2003.

[30] H. C. van Houwelingen, T. Bruinsma, A. A. Hart, L. J. van't Veer, and L. F. Wessels, "Cross-validated cox regression on microarray gene expression data," *Statistics in medicine*, vol. 25, no. 18, pp. 3201–3216, 2006.

[31] M. J. Van de Vijver, Y. D. He, L. J. van't Veer *et al.*, "A gene-expression signature as a predictor of survival in breast cancer," *New England Journal of Medicine*, vol. 347, no. 25, pp. 1999–2009, 2002.

[32] A. Rosenwald, G. Wright, W. C. Chan *et al.*, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma," *New England Journal of Medicine*, vol. 346, no. 25, pp. 1937–1947, 2002.

[33] D. G. Beer, S. L. Kardia, C.-C. Huang *et al.*, "Gene-expression profiles predict survival of patients with lung adenocarcinoma," *Nature medicine*, vol. 8, no. 8, pp. 816–824, 2002.

[34] L. Bullinger, K. Do¨hner, E. Bair, S. Fro¨hling, R. F. Schlenk, R. Tibshirani, H. Do¨hner, and J. R. Pollack, "Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia," *New England Journal of Medicine*, vol. 350, no. 16, pp. 1605–1616, 2004.

Liu Yang was born on December 30, 1987 in Hebei, China. He received a bachelor degree in electrical and computer engineering in 2010 from Shanghai Jiaotong University, and a M.Sc. degree in machine learning in 2011 from the University of Bristol.

He is now a Ph.D. student in Uppsala University, Department IT, Syscon, in Sweden since 2012.

Kristiaan Pelckmans was born on November 3, 1978 in Merksplas, Belgium. He received a M.Sc. degree (Licentiaat) in computer science in 2000 from the Katholieke Universiteit Leuven (K.U. Leuven), and a Ph.D. in applied mathematics in 2005 from K.U. Leuven.

He conducted (postdoctoral) research internationally e.g. in Darmstadt (DE), London (UK), and holds now an assistant professorship since 2009 in Uppsala University (SE), Department IT, Syscon.