Fast Training of Support Vector Machines for Survival Analysis

Sebastian Pölsterl[1](✉), Nassir Navab[1,2], and Amin Katouzian[1]

[1] Chair for Computer Aided Medical Procedures, Technische Universität München, Munich, Germany
{poelster,navab,katouzian}@in.tum.de

[2] Johns Hopkins University, Baltimore, MD, USA

Abstract. Survival analysis is a commonly used technique to identify important predictors of adverse events and develop guidelines for patient's treatment in medical research. When applied to large amounts of patient data, efficient optimization routines become a necessity. We propose efficient training algorithms for three kinds of linear survival support vector machines: 1) ranking-based, 2) regression-based, and 3) combined ranking and regression. We perform optimization in the primal using truncated Newton optimization and use order statistic trees to lower computational costs of training. We employ the same optimization technique and extend it for non-linear models too. Our results demonstrate the superiority of our proposed optimization scheme over existing training algorithms, which fail due to their inherently high time and space complexities when applied to large datasets. We validate the proposed survival models on 6 real-world datasets, and show that pure ranking-based approaches outperform regression and hybrid models.

Keywords: Survival analysis · Support vector machine · Optimization

1 Introduction

Recently, researchers have become interested in studying the effective use of electronic health records to improve outcomes of medical procedures, reduce health care costs, evaluate the efficiency of newly developed drugs, and predict health trends or adverse events (see e.g. [13] for an overview). In the latter case, survival analysis is employed to examine how a particular set of covariates affects the time until the occurrence of an event of interest, such as death or reaching a specific state of disease progression. The objective in survival analysis is to establish a connection between covariates and the time between the start of the study and an event. What makes survival analysis differ from traditional machine learning is the fact that parts of the training data can only be partially observed – they are *censored*. In a clinical study, patients are often monitored for a particular time period, and events occurring in this particular period are recorded. If a patient experiences an event, the exact time of the event can

A. Appice et al. (Eds.): ECML PKDD 2015, Part II, LNAI 9285, pp. 243–259, 2015.
DOI: 10.1007/978-3-319-23525-7_15

be recorded – the patient's record is *uncensored*. In contrast, *right censored* records refer to patients that remained event-free during the study period and it is unknown whether an event has or has not occurred after the study ended. Consequently, survival models demand for proper training algorithms that take this unique characteristic of such a dataset into account.

Cox's proportional hazards model [6] is the standard for analyzing time-to-event data, despite having several shortcomings: 1) it assumes that hazard functions for any two individuals are proportional, i.e., their ratio is constant over time, 2) it is not applicable to data with more features than samples, 3) it fails if features are highly correlated, and 4) its decision function is linear in the covariates. The advantage of large-margin methods for classification and regression has motivated researchers to adapt these models for survival analysis. Authors in [17,22] cast survival analysis as a regression problem and adapted support vector regression, whereas Eleuteri et al. [9] formulated a loss function derived from quantile regression. Steck et al. [23] observed that survival analysis can be expressed as a ranking problem, which led to extensions of Rank Support Vector Machines (RankSVMs) [10,24]. Finally, Van Belle et al. [26] proposed a hybrid solution between the ranking and regression approach.

The main disadvantage of ranking-based techniques is that their objective function depends on a quadratic number of constraints with respect to the number of training samples, which makes training intractable with medium to large sized datasets. By clustering data according to survival times, authors in [25] showed that the computational complexity can be lowered without considerable loss in performance. For regular RankSVMs, which do not account for censoring, authors in [2,19] proposed the use of order statistic trees to alleviate this problem.

In this paper, we extend the work of Lee et al. [19] to efficiently train ranking, and regression-based survival models by re-formulating their approach to be applicable to survival analysis in the presence of right censoring. In [10,24], ranking-based survival support vector machines were based on the hinge loss and optimization was carried out in the dual using a generic quadratic programming solver. In contrast, we use the squared hinge loss and perform truncated Newton optimization, which leads to a more efficient training algorithm. A further improvement is due to order statistic trees to avoid explicitly storing all pairwise comparisons of samples, which requires $O(n^2)$ space, where n is the number of samples. Moreover, we introduce a straightforward training technique for a combined regression and ranking approach. When considering non-linear functions, we demonstrate that training can still be carried out efficiently using the primal formulation. Finally, experimental results of 7 synthetic and 6 real world datasets justify the advantages of our proposed solution.

2 Survival Analysis

The objective in training a survival model is to derive a model's parameters in the presence of censoring. After training, the model can be used to predict the survival time of patients based on a given set of features. For a set of n patients, we know for the i-th patient: 1) the exact time $c_i \geq 0$ of censoring, i.e., the time until which the patient was observed, and 2) the time $t_i \geq 0$ when a patient experienced an event, if any. From these two quantities, we define the survival time y_i as

$$y_i = \min(t_i, c_i) = \begin{cases} t_i & \text{if } \delta_i = 1 \\ c_i & \text{if } \delta_i = 0, \end{cases}$$

where $\delta_i \in \{0, 1\}$ is the event indicator. Thus, training data for a survival model consists of triples $(\mathbf{x}_i, y_i, \delta_i)$, where $\mathbf{x}_i$ is a d-dimensional feature vector.

During training, information about the occurrence of an event is only partially available for censored patients, i.e., those that did not experience an event or dropped out of the study. When training a survival model, one has to consider that two patients i and j are only comparable if both experienced an event or only one of them experienced an event and the time of the event occurred before the time of censoring, formally: $(y_i < y_j \wedge \delta_i = 1) \vee (y_i > y_j \wedge \delta_j = 1)$. If two patients do not satisfy this condition, they are incomparable and their relation cannot be used to deduce a survival model.

Here, we discuss two approaches to survival analysis: the first approach treats survival analysis as a ranking problem, and the second approach as a regression problem. Finally, we present an objective function that combines both ideas. Our implementation of the methods proposed in this paper are publicly available.[1]

3 Survival Analysis as Ranking Problem

In ranking, the goal is to recover the correct order of samples according to their relevance. For survival analysis, relevance corresponds to the survival time. However, not all pairwise comparisons are meaningful in the presence of right censoring. The set $\mathcal{P} = \{(i, j) \mid y_i > y_j \wedge \delta_j = 1\}_{i,j=1,\ldots,n}$ defines the pairs of comparable samples that can be used for training and $p = |\mathcal{P}|$ the cardinality of this set, which is bounded by $O(n^2)$. We minimize our objective function similar to the work in [19], but additionally account for right censoring during training.

Definition 1. *The objective function of ranking-based linear survival support vector machine is defined as*

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2} \sum_{i,j \in \mathcal{P}} \max(0, 1 - (\boldsymbol{w}^T\boldsymbol{x}_i - \boldsymbol{w}^T\boldsymbol{x}_j))^2, \qquad (1)$$

where $\boldsymbol{w} \in \mathbb{R}^d$ are the coefficients and $\gamma > 0$ is a regularization parameter. A new set of data points $\boldsymbol{X}_{\text{new}}$, can be ranked with respect to their predicted survival time according to elements of $\boldsymbol{X}_{\text{new}}\boldsymbol{w}$.

[1] https://github.com/tum-camp/survival-support-vector-machine

The sum in the second term of (1) has a complexity of $O(n^2)$ and thus training with only a few thousand samples is already intractable. We will first derive a gradient-based minimization of the objective function, based on Newton's method, and then outline a more efficient optimization, which does not depend on the number of comparable pairs, using truncated Newton optimization and order statistic trees.

The objective function (1) can be expressed in matrix form as

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\left(\mathbb{1} - \boldsymbol{A}\boldsymbol{X}\boldsymbol{w}\right)^T \boldsymbol{D}_{\boldsymbol{w}}\left(\mathbb{1} - \boldsymbol{A}\boldsymbol{X}\boldsymbol{w}\right), \tag{2}$$

where $\mathbb{1}$ is a vector of all ones, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^T$, and $\boldsymbol{A} \in \mathbb{R}^{p \times n}$ a sparse matrix with $A_{ki} = 1$ and $A_{kj} = -1$ if $(i,j) \in \mathcal{P}$ and zero otherwise. $\boldsymbol{D}_{\boldsymbol{w}}$ is a $p \times p$ diagonale matrix that has an entry for each $(i,j) \in \mathcal{P}$ that indicates whether this pair is a support vector, i.e., $1 - (\boldsymbol{w}^T\boldsymbol{x}_i - \boldsymbol{w}^T\boldsymbol{x}_j) > 0$ [19]. For the k-th item of $\mathcal{P}$, representing the pair (i,j), the corresponding entry in $\boldsymbol{D}_{\boldsymbol{w}}$ is defined as

$$(\boldsymbol{D}_{\boldsymbol{w}})_{k,k} = \begin{cases} 1 & \text{if } \boldsymbol{w}^T\boldsymbol{x}_j > \boldsymbol{w}^T\boldsymbol{x}_i - 1 \\ 0 & \text{else} \end{cases}. \tag{3}$$

Thus, we obtain an objective function that is convex in $\boldsymbol{w}$ and can apply Newton's method to minimize it. One update in Newton's method with step size μ becomes

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w} - \mu\left(\frac{\partial^2 f}{\partial\boldsymbol{w}\partial\boldsymbol{w}^T}\right)^{-1}\frac{\partial f}{\partial\boldsymbol{w}} \tag{4}$$

with partial derivatives

$$\frac{\partial f}{\partial\boldsymbol{w}} = \boldsymbol{w} + \gamma\boldsymbol{X}^T\left(\boldsymbol{A}^T\boldsymbol{D}_{\boldsymbol{w}}\boldsymbol{A}\boldsymbol{X}\boldsymbol{w} - \boldsymbol{A}^T\boldsymbol{D}_{\boldsymbol{w}}\mathbb{1}\right) \tag{5}$$

$$\frac{\partial^2 f}{\partial\boldsymbol{w}\partial\boldsymbol{w}^T} = \boldsymbol{I} + \gamma\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{D}_{\boldsymbol{w}}\boldsymbol{A}\boldsymbol{X}. \tag{6}$$

Note that we used the generalized Hessian in the second derivative, because $f(\boldsymbol{w})$ is not twice differentiable at $\boldsymbol{w}$ [16].

Next, we simplify the derivatives by expressing the product $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{D}_{\boldsymbol{w}}\boldsymbol{A}$ in terms of a new matrix $\boldsymbol{A}_{\boldsymbol{w}} \in \{-1, 0, 1\}^{p_{\boldsymbol{w}}, n}$ that is a restricted version of $\boldsymbol{A}$, limited to rows corresponding to support vectors:

$$\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{D}_{\boldsymbol{w}}\boldsymbol{A} = \boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}, \tag{7}$$

where $p_{\boldsymbol{w}}$ denotes the number of pairs $(i,j) \in \mathcal{P}$ – rows of $\boldsymbol{A}$ – where $\boldsymbol{w}^T\boldsymbol{x}_j > \boldsymbol{w}^T\boldsymbol{x}_i - 1$.

Algorithm 1. Survival Support Vector Machine Training.

```
Input: Training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i, \delta_i)\}_{i=1}^n$, hyper-parameter $\gamma > 0$.
Output: Coefficients $\boldsymbol{w}$.
1  Randomly resolve ties in survival times $y_i$ $\forall i \in \{1, \ldots, n\}$;
2  $\boldsymbol{w}^0 \leftarrow \boldsymbol{0}$;
3  $t \leftarrow 0$;
4  while not converged do
5      Use conjugate gradient to determine search direction $\boldsymbol{u} = \left(\frac{\partial^2 f}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T}\right)^{-1} \frac{\partial f}{\partial \boldsymbol{w}}$
       with $\boldsymbol{w} = \boldsymbol{w}^t$;
6      Choose step size $\mu$ by backtracking line search;
7      Update $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \mu \boldsymbol{u}$;
8      $t \leftarrow t + 1$;
9  end
10 $\boldsymbol{w} \leftarrow \boldsymbol{w}^t$;
```

Definition 2. *Formula (2) and its derivatives can be re-formulated using $\boldsymbol{A_w}$ to eliminate $\boldsymbol{D_w}$.*

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\left(p_w + \boldsymbol{w}^T\boldsymbol{X}^T\left(\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{A}_{\boldsymbol{w}}^T\mathbb{1}\right)\right) \tag{8}$$

$$\frac{\partial f}{\partial \boldsymbol{w}} = \boldsymbol{w} + \gamma\boldsymbol{X}^T\left(\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{X}\boldsymbol{w} - \boldsymbol{A}_{\boldsymbol{w}}^T\mathbb{1}\right) \tag{9}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{w}\partial \boldsymbol{w}^T} = \boldsymbol{I} + \gamma\boldsymbol{X}^T\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{X} \tag{10}$$

3.1 Truncated Newton Optimization

Medical research is often challenging due to high-dimensional data: a patient's health record comprises several hundred features, and microarray data consists of several thousand measurements. In this applications, explicitly computing and storing the Hessian matrix can be prohibitive, therefore, we use a truncated Newton method that uses a linear conjugate gradient method to compute the search direction [7,16,20]. This only requires the computation of the Hessian-vector product $\boldsymbol{Hv}$, which can be computed by

$$\boldsymbol{Hv} = \boldsymbol{v} + \gamma\boldsymbol{X}^T\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{X}\boldsymbol{v}. \tag{11}$$

Thus, the complexity of a single conjugate gradient iteration is $O(nd + p + d)$, when multiplying from the right, which is lower than $O(pd^2 + pd + d)$ to obtain the full Hessian matrix. Truncated Newton optimization consists of an outer loop to update the coefficients $\boldsymbol{w}$ and an inner loop to find the search direction via conjugate gradient (see algorithm 1).

3.2 Efficient Calculation of Search Direction

In each iteration of Newton's method, $\boldsymbol{A_w}$ has to be recomputed due to its dependency on $\boldsymbol{w}$, which requires iterating over all comparable pairs, being of order $\binom{n}{2}$. Therefore, the complexity of learning a new model is still quadratic in the number of samples. Next, we will derive an improved algorithm that avoids constructing $\boldsymbol{A_w}$ explicitly. First, we derive the conditions under which an entry in $\boldsymbol{A_w}$ is non-zero, followed by proposing a compact representation of an entry in $\boldsymbol{A_w^T A_w}$, which finally leads to an efficient optimization scheme that is independent of the size of $\mathcal{P}$.

Proposition 1. *For $k \in \{1, \ldots, p_w\}$ and $q \in \{1, \ldots, n\}$, $(\boldsymbol{A_w})_{k,q} = 1$ if all of the following conditions are satisfied:*

(a) survival time of q-th sample is lower *than survival time of some sample $s \in \{1, \ldots, n\}$ (s outlives q): $y_q < y_s$.*
(b) the q-th sample is uncensored: $\delta_q = 1$.
(c) the pair $(s, q) \in \mathcal{P}$ is a support vector: $\boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_q + 1$.

Proposition 2. *For $k \in \{1, \ldots, p_w\}$ and $q \in \{1, \ldots, n\}$, $(\boldsymbol{A_w})_{k,q} = -1$ if all of the following conditions are satisfied:*

(a) survival time of q-th sample is higher *than survival time of some sample $s \in \{1, \ldots, n\}$ (q outlives s): $y_q > y_s$.*
(b) the s-th sample is uncensored: $\delta_s = 1$.
(c) the pair $(q, s) \in \mathcal{P}$ is a support vector: $\boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_q - 1$.

Proof. Note that the only difference between both propositions is the order of samples s and q with respect to their survival times. Thus, the first proposition can be transformed into the second by swaping s and q, and vice versa. Conditions (a) and (b) are directly derived from the definition of $\boldsymbol{A}$. Each row of $\boldsymbol{A}$ and $\boldsymbol{A_w}$ contains exactly one element that is 1, one element that is -1, and the rest is all zeros. For each pair of samples (row of $\boldsymbol{A}$), the sample with the shorter survival time is assigned 1, and the other sample -1, which is reflected by condition (a). In addition, each pair must be comparable, i.e., the sample with the shorter survival time must be uncensored, which leads to condition (b). Finally, condition (c) is due to the multiplication $\boldsymbol{AD_w}$ that restricts rows of $\boldsymbol{A}$ to pairs of samples that are support vectors. □

If proposition 1 or 2 holds, the result of the multiplication $(\boldsymbol{A_w})_{k,i} \cdot (\boldsymbol{A_w})_{k,j}$ is either 1 or -1, if $i = j$ or $i \neq j$, respectively, for $k \in \{1, \ldots, p_w\}$ and $i, j \in \{1, \ldots, n\}$. In the latter case, the conditions of propositions 1 and 2 are equal.

Combining all cases, the product $(\boldsymbol{A_w})_{k,i} \cdot (\boldsymbol{A_w})_{k,j}$ is defined as

$$(\boldsymbol{A_w})_{k,i} \cdot (\boldsymbol{A_w})_{k,j} = \begin{cases} 1 & \text{if } i = j,\ (\boldsymbol{A_w})_{k,i} = (\boldsymbol{A_w})_{k,j} = 1, \\ & \quad \text{and proposition 1 holds for } q = i, \\ 1 & \text{if } i = j,\ (\boldsymbol{A_w})_{k,i} = (\boldsymbol{A_w})_{k,j} = -1, \\ & \quad \text{and proposition 2 holds for } q = i, \\ -1 & \text{if } i \neq j,\ (\boldsymbol{A_w})_{k,i} = 1, (\boldsymbol{A_w})_{k,j} = -1, \\ & \quad \text{and proposition 1 holds for } q = i, s = j \\ & \quad \Leftrightarrow \text{ proposition 2 holds for } q = j, s = i, \\ -1 & \text{if } i \neq j,\ (\boldsymbol{A_w})_{k,i} = -1, (\boldsymbol{A_w})_{k,j} = 1, \\ & \quad \text{and proposition 1 holds for } q = j, s = i, \\ & \quad \Leftrightarrow \text{ proposition 2 holds for } q = i, s = j, \\ 0 & \text{else.} \end{cases} \tag{12}$$

We can compactly express $\left(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w}\right)_{i,j} = \sum_{k=1}^{p_{\boldsymbol{w}}} (\boldsymbol{A_w})_{k,i} \cdot (\boldsymbol{A_w})_{k,j}$ using above definitions and by defining the following two sets and their cardinalities.

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_i + 1 \wedge \delta_i = 1\} \qquad l_i^+ = |\mathrm{SV}_i^+| \tag{13}$$

$$\mathrm{SV}_i^- = \{s \mid y_s < y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_i - 1 \wedge \delta_s = 1\} \qquad l_i^- = |\mathrm{SV}_i^-| \tag{14}$$

The set SV_i^+ represents proposition 1, and SV_i^- represents proposition 2. This allows us to compactly express an entry of $\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w}$ as

$$(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w})_{i,j} = \begin{cases} l_i^+ + l_i^- & \text{if } i = j, \\ -1 & \text{if } i \neq j, \text{ and } j \in \mathrm{SV}_i^+ \text{ or } j \in \mathrm{SV}_i^-, \\ 0 & \text{else,} \end{cases} \tag{15}$$

where the second case is due to only one addend being non-zero, because each pair of samples is compared only once.

The term $\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w X v}$ is part of the objective function, its gradient, and the Hessian-vector product. Applying the formulation in (15), we obtain

$$\begin{aligned} (\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w X v})_i &= (l_i^+ + l_i^-) \boldsymbol{x}_i^T \boldsymbol{v} - \sum_{s \in \mathrm{SV}_i^+} \boldsymbol{x}_s \boldsymbol{v} - \sum_{s \in \mathrm{SV}_i^-} \boldsymbol{x}_s \boldsymbol{v} \\ &= (l_i^+ + l_i^-) \boldsymbol{x}_i^T \boldsymbol{v} - \sigma_i^+ - \sigma_i^- . \end{aligned} \tag{16}$$

and

$$\boldsymbol{X}^T \boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A_w X v} = \boldsymbol{X}^T \begin{pmatrix} (l_1^+ + l_1^-) \boldsymbol{x}_1^T \boldsymbol{v} - (\sigma_1^+ + \sigma_1^-) \\ \vdots \\ (l_n^+ + l_n^-) \boldsymbol{x}_n^T \boldsymbol{v} - (\sigma_n^+ + \sigma_n^-) \end{pmatrix} . \tag{17}$$

Additionally, the objective function and its gradient contain the term $\boldsymbol{A}_{\boldsymbol{w}}^T \mathbb{1}$, where one component is computed as

$$\begin{aligned}(\boldsymbol{A}_{\boldsymbol{w}}^T \mathbb{1})_i &= |\mathrm{SV}_i^+ \cup \mathrm{SV}_i^-| \\ &= |\{(s,t) \mid y_t < y_i < y_s \wedge \delta_t = 1 \wedge \delta_i = 1 \wedge \\ &\qquad \boldsymbol{w}^T\boldsymbol{x}_s - 1 < \boldsymbol{w}^T\boldsymbol{x}_i < \boldsymbol{w}^T\boldsymbol{x}_t + 1\}| \\ &= l_i^- - l_i^+ .\end{aligned} \tag{18}$$

By substituting (17) and (18) together with $p_{\boldsymbol{w}} = \sum_{i=1}^n l_i^+ = \sum_{i=1}^n l_i^-$ into (8), (9), and (11), all terms that depend on $\boldsymbol{A}_{\boldsymbol{w}}$ during optimization can be eliminated. Assuming that l_i^+, l_i^-, σ_i^+, and σ_i^- have been computed already, the complexity of evaluating the objective function, gradient, and Hessian-vector product is $O(nd+d)$. Subsequently, we will discuss an efficient method to obtain these values.

3.3 Improving Optimization by Order Statistic Trees

The main difficulty is that the order of actual survival times y_i and predictions $\boldsymbol{w}^T\boldsymbol{x}_i$ have to be considered when constructing the sets SV_i^+ and SV_i^-. Assuming that samples have been sorted in ascending order according to $\boldsymbol{w}^T\boldsymbol{x}_i$, we illustrate how both sets can be constructed by the following example:

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0

As we can see, the first element for which $\mathrm{SV}_i^+ \neq \varnothing$ occurs at $i = 3$, because both the first and second sample are censored ($\delta_i = 0$), which violates condition (b) of proposition 1. For $i = 3$, we obtain $\mathrm{SV}_3^+ = \{s | y_s > 6 \wedge \boldsymbol{w}^T\boldsymbol{x}_s < 1.15\} = \{2, 5\}$. The next set ($i = 4$) is again empty, because of censoring, and $\mathrm{SV}_5^+ = \{s | y_s > 8 \wedge \boldsymbol{w}^T\boldsymbol{x}_s < 1.3\} = \{2\}$. This example shows, that SV_i^+ is non-empty if and only if the i-th sample is uncensored, and that SV_{i+1}^+ can be constructed incrementally from the set SV_i^+:

$$\begin{aligned}&\{s | \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_{i+1} + 1 \wedge \delta_{i+1} = 1\} \\ =&\{s | \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1\} \cup \{s | \boldsymbol{w}^T\boldsymbol{x}_i + 1 \le \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_{i+1} + 1 \wedge \delta_{i+1} = 1\}.\end{aligned}$$

When constructing the set SV_i^-, we can obtain a similar incremental update rule when iterating the list of samples according to decreasing values of $\boldsymbol{w}^T\boldsymbol{x}_i$. Here, $\mathrm{SV}_9^- = \varnothing$, because no element with $\boldsymbol{w}^T\boldsymbol{x}_s > 1.3$ satisfies conditions (a) and (b) of proposition 2, and $\mathrm{SV}_8^- = \{s | y_s < 3 \wedge \boldsymbol{w}^T\boldsymbol{x}_s > 0.7 \wedge \delta_s = 1\} = \{6\}$. An incremental update when going from i to $i-1$ is defined as

$$\begin{aligned}\{s | \boldsymbol{w}^T\boldsymbol{x}_s > \boldsymbol{w}^T\boldsymbol{x}_{i-1} - 1 \wedge \delta_s = 1\} &= \{s | \boldsymbol{w}^T\boldsymbol{x}_s > \boldsymbol{w}^T\boldsymbol{x}_i - 1 \wedge \delta_s = 1\} \\ &\cup \{s | \boldsymbol{w}^T\boldsymbol{x}_i - 1 \ge \boldsymbol{w}^T\boldsymbol{x}_s > \boldsymbol{w}^T\boldsymbol{x}_{i-1} - 1 \wedge \delta_s = 1\}.\end{aligned}$$

To maintain the respective sets of relevant samples for computing SV_i^+ and SV_i^-, we incrementally add elements y_i and $\boldsymbol{x}_i^T \boldsymbol{v}$ to an order statistic tree that allows retrieving $|\{s | y_s > y_i\}|$ and $|\{s | y_s < y_i\}|$ in logarithmic time. Note that both sets in the incremental update of SV_i^- consider censoring, whereas for SV_i^+ censoring is only relevant for the second set, but not the first. For the former, we use an order statistic tree to sort *uncensored* samples according to their survival time y_i, and for the latter we sort *all* samples, disregarding censoring. Formally, an order statistic tree is defined as follows.

Definition 3. *An order statistic tree is a balanced binary search tree that stores key-value pairs and has the following properties.*

1. *For an internal node x with left child* $\mathrm{left}(x)$ *and right child* $\mathrm{right}(x)$*:*

$$\mathrm{key}(\mathrm{left}(x)) \leq \mathrm{key}(x) \textit{ and } \mathrm{key}(\mathrm{right}(x)) \geq \mathrm{key}(x).$$

2. *For n elements in the tree, the height of the tree is limited by $O(\log n)$.*
3. *Each node x in the tree stores two additional attributes size and sum.*
   (a) *size denotes the size of the subtree mounted at x:*

$$\mathrm{size}(x) = \begin{cases} 0 & \textit{if } x = \varnothing \\ \mathrm{size}(\mathrm{left}(x)) + \mathrm{size}(\mathrm{right}(x)) + 1 & \textit{else} \end{cases}$$

   (b) *sum denotes the sum of all values in the subtree mounted at x:*

$$\mathrm{sum}(x) = \begin{cases} 0 & \textit{if } x = \varnothing \\ \mathrm{sum}(\mathrm{left}(x)) + \mathrm{sum}(\mathrm{right}(x)) + \mathrm{value}(x) & \textit{else} \end{cases}$$

4. *The correct value for above attributes is maintained after insertion.*

Based on aforementioned definitions, we use algorithm 2 to compute l_i^+, xv_i^+, l_i^- and xv_i^-. The auxiliary function `CountSmaller` is defined in algorithm 3, and `CountLarger` works in a similar manner. The complexity of these functions corresponds to the complexity of finding an element in a binary search tree, which is $O(\log n)$. Hence, the overall complexity of algorithm 2 is $O(n \log n)$, and the Hessian-vector product in (11) can be carried out in $O(nd + d + n \log n)$, after sorting according to $\boldsymbol{w}^T \boldsymbol{x}_i$, which costs $O(n \log n)$. Thus, one conjugate gradient iteration does not depend on the number of comparable pairs p anymore, which scales quadratically in the number of samples. Finally, the overall complexity of training a ranking-based survival support vector machine as outlined in algorithm 1 is

$$[O(n \log n) + O(nd + d + n \log n)] \cdot \bar{N}_{\mathrm{CG}} \cdot N_{\mathrm{Newton}}, \tag{19}$$

where $\bar{N}_{\mathrm{CG}}$ and N_{Newton} are the average number of conjugate gradient iterations and the total number of Newton updates, respectively.

Algorithm 2. Efficient computation of l_i^+, l_i^-, σ_i^+, and σ_i^-.

```
Input: Training data $\mathcal{D} = \{(\boldsymbol{x}_k, y_k, \delta_k)\}_{k=1}^n$, coefficient vectors $\boldsymbol{w}$ and $\boldsymbol{v}$.
Output: $l_i^+$, $l_i^-$, $\sigma_i^+$, and $\sigma_i^-$ $\forall i \in \{1, \ldots, n\}$
1  Sort all $\boldsymbol{w}^T\boldsymbol{x}_i$ in ascending order, such that $\boldsymbol{w}^T\boldsymbol{x}_{\pi(1)} \leq \cdots \leq \boldsymbol{w}^T\boldsymbol{x}_{\pi(n)}$;
2  $T \leftarrow$ an empty order statistic tree;
3  $j \leftarrow 1$;
4  for $i \leftarrow 1$ to $n$ do
5      while $j \leq n$ and $\boldsymbol{w}^T\boldsymbol{x}_{\pi(j)} < \boldsymbol{w}^T\boldsymbol{x}_{\pi(i)} + 1$ do
6          Insert $(y_{\pi(j)}, \boldsymbol{x}_{\pi(j)}^T\boldsymbol{v})$ into $T$;
7          $j \leftarrow j + 1$;
8      end
9      if $\delta_{\pi(i)} = 1$ then
10         $(l_{\pi(i)}^+, xv_{\pi(i)}^+) \leftarrow$ CountLarger(root of $T$, $y_{\pi(i)}$);
11     else
12         $(l_{\pi(i)}^+, xv_{\pi(i)}^+) \leftarrow (0, 0)$;
13     end
14 end
15 $j \leftarrow n$;
16 $T \leftarrow$ an empty order statistic tree;
17 for $i \leftarrow n$ to $1$ do
18     while $j \geq 1$ and $\boldsymbol{w}^T\boldsymbol{x}_{\pi(j)} > \boldsymbol{w}^T\boldsymbol{x}_{\pi(i)} - 1$ do
19         if $\delta_{\pi(j)} = 1$ then Insert $(y_{\pi(j)}, \boldsymbol{x}_{\pi(j)}^T\boldsymbol{v})$ into $T$;
20         $j \leftarrow j - 1$;
21     end
22     $(l_{\pi(i)}^-, xv_{\pi(i)}^-) \leftarrow$ CountSmaller(root of $T$, $y_{\pi(i)}$);
23 end
```

Algorithm 3. CountSmaller

```
Input: node $x$ in order statistic tree, survival time $y_i$
Output: $l_i^-$ (number of uncensored samples with $y_s < y_i$), and
        $\sigma_i^- = \sum_{s \in \mathrm{SV}_i^-} \boldsymbol{x}_i^T\boldsymbol{v}$
1  if $x = \varnothing$ then
2      $l_i^- \leftarrow 0; \sigma_i^- \leftarrow 0$;
3  else if key$(x) = y_i$ then
4      $l_i^- \leftarrow$ size(left$(x)$);
5      $\sigma_i^- \leftarrow$ sum(left$(y)$);
6  else if key$(x) < y_i$ then
7      $(l_i^-, \sigma_i^-) \leftarrow$ CountSmaller(right$(x), y_i$);
8      $l_i^- \leftarrow l_i^- +$ size$(x) -$ size(right$(x)$);
9      $\sigma_i^- \leftarrow \sigma_i^- +$ sum$(x) -$ sum(right$(x)$);
10 else // key$(x) > y_i$
11     $(l_i^-, \sigma_i^-) \leftarrow$ CountSmaller(left$(x), y_i$);
12 end
```

4 Survival Analysis as Regression Problem

Instead of treating survival analysis as a ranking problem, authors have proposed regression-based approaches using an absolute loss as well [17,22]. In contrast to a ranking-based model, a regression model can predict the exact time of an event. Training algorithms for such a model need to be aware of censored patient record as well. For right censored patients – those who did not experience an event – no information about the correctness of predicted survival times beyond the time of censoring is available. A valid error can only be computed for patients that experienced an event during the study period, or if the predicted survival time is too early, i.e., before the time of censoring. Experiments in [26] revealed that survival models based on ε-insensitive support vector regression worked equally well if the insensitive zone is set to zero. Hence, our regression objective is based on an ordinary least square problem with ℓ_2 penalty and the additional consideration of right censoring.

$$f_{\text{Regr.}}(\boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\sum_{i=0}^{n}\left(\zeta_{\boldsymbol{w},b}(y_i, x_i, \delta_i)\right)^2 \tag{20}$$

$$\zeta_{\boldsymbol{w},b}(y_i, \boldsymbol{x}_i, \delta_i) = \begin{cases} \max(0, y_i - \boldsymbol{w}^T\boldsymbol{x}_i - b) & \text{if } \delta_i = 0, \\ y_i - \boldsymbol{w}^T\boldsymbol{x}_i - b & \text{if } \delta_i = 1, \end{cases} \tag{21}$$

where $b \in \mathbb{R}$ is the intercept.

By combining all parameters into a single vector $\boldsymbol{\omega} = (b, \boldsymbol{w})^T$, and extending $\boldsymbol{X}$ by a column of all ones to accommodate the intercept, the objective can be expressed in matrix form as follows:

$$f_{\text{Regr.}}(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + \frac{\gamma}{2}\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega}\right)^T \boldsymbol{R}_{\boldsymbol{\omega}}\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega}\right) \tag{22}$$

where $\boldsymbol{R}_{\boldsymbol{\omega}}$ is a diagonal matrix with the i-th element being 1 if $y_i > \boldsymbol{w}^T\boldsymbol{x}_i + b$ or $\delta_i = 1$, and zero otherwise. Due to $f_{\text{Regr.}}$ being a convex quadratic function, we can use truncated Newton optimization to minimize it, as described in algorithm 1. In addition, we can easily create a hybrid model that addresses the ranking and regression objective concurrently; its objective function is defined as

$$f_{\text{hybrid}}(\boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\left[\alpha \sum_{i,j \in \mathcal{P}} \max(0, 1 - (\boldsymbol{w}^T\boldsymbol{x}_i - \boldsymbol{w}^T\boldsymbol{x}_j))^2 + (1-\alpha)\sum_{i=0}^{n}\left(\zeta_{\boldsymbol{w},b}(y_i, x_i, \delta_i)\right)^2\right]. \tag{23}$$

The hyper-parameter $\alpha \in [0, 1]$ controls the relative weight of the regression and ranking objective. Clearly, if $\alpha = 1$ it reduces to the ranking objective, and if $\alpha = 0$ to the regression objective.

5 Non-linear Extension

So far, we only discussed linear survival support vector machines and their efficient training in the primal. If data are more complex, one might want to model non-linear functions through the use of kernel functions. Commonly, the representer theorem [18] is employed and optimization is carried out in the dual rather than the primal. The weights $\boldsymbol{w}$ are then a linear combination of the training samples. However, if training data is large, the number of support vectors increases as well, resulting in excessive computational costs. Chapelle et al. [5] showed that solving the non-linear problem is equivalent to the combination of Kernel PCA and training in the primal. Thus, efficient training of non-linear survival models is straightforward using the optimization scheme outlined above.

6 Experiments

In our experiments, we first studied the efficiency of our proposed algorithm to minimize the ranking-based objective function and then investigated the predictive performance of ranking, regression, and hybrid approaches. We standardized continuous features to have zero mean and unit standard deviation, and randomly resolved ties in survival times before optimization. For regression, we used the logarithm of survival times y_i as target value.

6.1 Computational Efficiency

In the first set of experiments, we compared the training time of three different formulations of the ranking-based objective function: the simple formulation in (2), the alternative formulation in (8), and our efficient proposed formulation in (17). We generated synthetic survival data of varying size following [4]. Data consisted of 10 normal distributed features and two redundant features, which were linear combinations of a subset of the first ten features. Correlations between the first ten features were defined as follows: $r_{1,3} = 0.03$, $r_{2,5} = 0.42$, $r_{3,5} = 0.08$, $r_{3,9} = 0.03$, $r_{5,8} = -0.55$, $r_{6,9} = 0.32$, and the remainder all zero. Survival times were Gompertz distributed and depended on a linear combination of all features. Finally, half of the samples were randomly censored. Our choice of order statistic trees were red-black trees [3] and AVL trees [1]. To minimize the influence of the operating system's process scheduler in our measurements, we report the lowest training time of ten repetitions in wall time.

Figure 1 shows the lowest training time following algorithm 1. The naive and improved optimization failed with more than 20,000 samples because of excessive memory requirements due to explicitly constructing the sparse matrix $\boldsymbol{A}$ and $\boldsymbol{A_w}$, respectively. For all datasets, optimization converged after less than 20 iterations. Although $\boldsymbol{A}$ has to be constructed only once for the simple optimization, training time quickly degenerates because it repeatedly has to be multiplied by $\boldsymbol{Xw}$, which takes $O(pn)$ time. The improved optimization updates $\boldsymbol{A_w}$ after

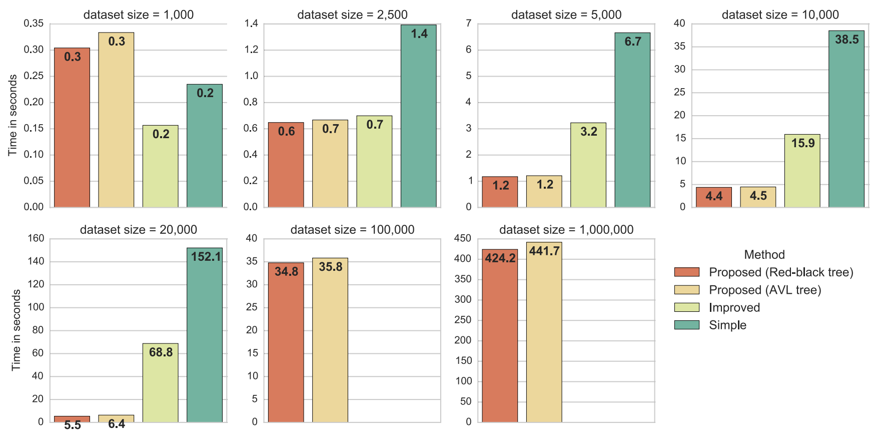

Fig. 1. Training time of survival models using ranking objective with truncated Newton optimization. Simple refers to the objective function in (2) and improved to the one in (8). Our proposed algorithm uses the efficient formulation in (17) with red-black trees or AVL trees.

Table 1. Overview of datasets used in our experiments.

Dataset	n	d	Events	Outcome
AIDS study [12]	1,151	13	96 (8.3%)	AIDS defining event or death
Breast cancer [8]	198	80	62 (31.3%)	Distant metastases
Coronary artery disease [21]	1,204	60	196 (15.9%)	Myocardial infarction or death
Framingham Offspring [15]	4,892	150	1,166 (23.8%)	Coronary vessel disease
Veteran's Lung Cancer [14]	137	6	128 (93.4%)	Death
Worcester Heart Attack Study [12]	500	14	215 (43.0%)	Death

each iteration of Newton's method, but only needs to perform $O(p_{\boldsymbol{w}}n)$ operations when multiplied by $\boldsymbol{Xw}$, which results in a lower training time. Using order statistic trees, the training time and memory requirements can be lowered significantly; for very large datasets, red-black trees were superior to AVL trees.

6.2 Prediction Performance

We evaluated the predictive performance of our proposed method for survival analysis on six real-world datasets of varying size, number of features, and amount of censoring (see table 1). In addition to the three models proposed here, we included Cox's proportional hazards model [6] with ℓ_2 (ridge) penalty, and ranking-based survival SVM with hinge loss [10, 24]. The regularization parameter γ for survival SVM controls the weight of the (squared) hinge loss, whereas for Cox's proportional hazards model, $\lambda = \gamma^{-1}$ controls the weight of the ℓ_2 penalty.

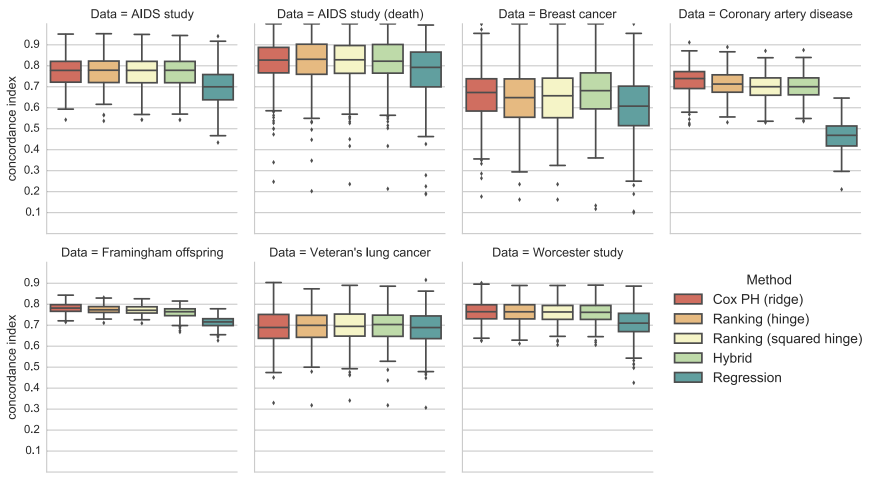

Fig. 2. Concordance index of Cox's proportional hazards model with ℓ_2 (ridge) penalty and four different survival SVM models: ranking objective with hinge loss, ranking objective with squared hinge loss, regression objective, and combined ranking and regression (hybrid).

Optimal performance was determined by a grid search over hyper-parameters. We set γ and λ to 2^i, where we altered i from -12 to 12 in steps of 2. Similar for α, which ranged from 0.05 to 0.95 in steps of 0.05. The maximum number of iterations of Newton's method was one thousand. Performance was measured by Harrell's concordance index (c index) [11], which is the ratio of correctly ordered pairs to comparable pairs. A c index of 0.5 corresponds to a random model and 1.0 to a perfect model. In addition, we measured the root mean squared error (RMSE) on uncensored patients to evaluate regression models. For each parameter setting, we randomly split each dataset into two equally sized parts, one for training and one for testing. Results reported here are with respect to the configuration that performed best on the training portion of 200 different random splits.

Figure 2 summarizes the results of our experiments with respect to c index. We observed that ranking-based approaches to survival analysis, using hinge or squared hinge loss, were comparable to Cox's proportional hazards model with ℓ_2 penalty and superior to a regression-based approach. We believe this is why the combined ranking-regression technique did not exceed the performance of the pure ranking approach. In fact, hyper-parameter search assigned more weight to the ranking objective in all cases but one. The only exception occurred for the breast cancer dataset, where $\alpha = 0.25$ was chosen and the hybrid model performed best. The reason for this becomes obvious when looking at the RMSE shown in figure 3. Predictions of survival time are off by a large extent on all datasets, which renders the regression objective unsuitable. This can be explained by the distribution of survival times, which are – even

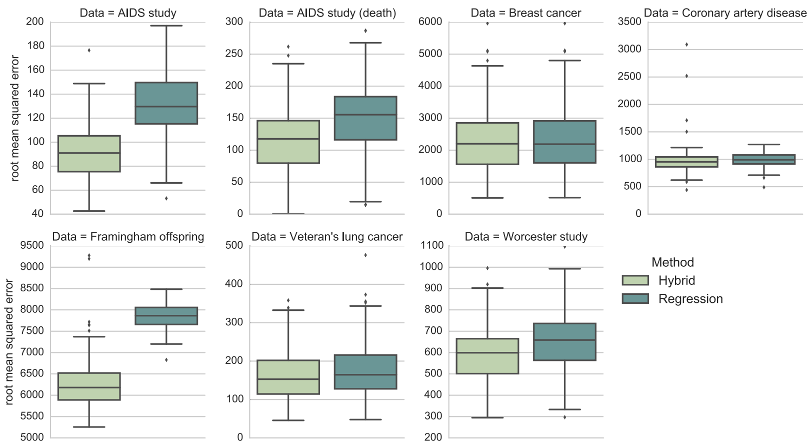

Fig. 3. Root mean squared error (RMSE) of regression-based and hybrid survival support vector machine.

after log-transformation – far from normally distributed, and thus violate a basic assumption of ordinary least squares. In [26] however, regression was based on absolute loss and outperformed ranking. A possible explanation might be the fact that squared loss is more sensitive to outliers than absolute loss. This problem could be alleviated by introducing sample weights to reduce the influence of outliers in the squared loss function. Finally, the performance of all approaches varied to a similar degree among 200 randomly selected train-test splits. We obtained similar results for non-linear survival models.

7 Conclusion

In this paper, we proposed an efficient method for training ranking-based and regression-based survival support vector machines. Our algorithm accounts for right censoring of patient records and avoids explicitly constructing a matrix of pairwise constraints – quadratic in the number of samples – by using order statistic trees. We experimentally showed that the reduced time and space complexity allow efficient training of survival models based on millions of patients, which would otherwise not been possible on commodity hardware. In addition to its high efficiency, the algorithm can be easily adapted for training non-linear as well as hybrid ranking and regression survival models. This opens up the opportunity to build survival models from large sets of medical health records to obtain new insights about the impact of particular factors on a disease.

Acknowledgments. We thank the Leibniz Supercomputing Centre (LRZ, www.lrz.de) for providing the computational resources for our experiments. Data were provided by the Framingham Heart Study of the National Heart Lung and Blood Institute of the National Institutes of Health and Boston University School of Medicine (Contract No.N01-HC-25195).

References

1. Adelson-Velsky, G., Landis, E.: An algorithm for the organization of information. In: Doklady Akademii Nauk SSSR, vol. 146, pp. 263–266 (1962)
2. Airola, A., Pahikkala, T., Salakoski, T.: Training linear ranking SVMs in linearithmic time using red–black trees. Pattern Recogn. Lett. **32**(9), 1328–1336 (2011)
3. Bayer, R.: Symmetric binary B-trees: Data structure and maintenance algorithms. Acta Inform. **1**(4), 290–306 (1972)
4. Bender, R., Augustin, T., Blettner, M.: Generating survival times to simulate Cox proportional hazards models. Stat. Med. **24**(11), 1713–1723 (2005)
5. Chapelle, O., Keerthi, S.S.: Efficient algorithms for ranking with SVMs. Information Retrieval **13**(3), 201–215 (2009)
6. Cox, D.R.: Regression models and life tables (with discussion). J. Roy. Stat. Soc. B **34**, 187–220 (1972)
7. Dembo, R.S., Steihaug, T.: Truncated Newton algorithms for large-scale optimization. Math. Programming **26**(2), 190–212 (1983)
8. Desmedt, C., Piette, F., Loi, S., Wang, Y., Lallemand, F., Haibe-Kains, B., Viale, G., Delorenzi, M., Zhang, Y., d'Assignies, M.S., Bergh, J., Lidereau, R., Ellis, P., Harris, A.L., Klijn, J.G., Foekens, J.A., Cardoso, F., Piccart, M.J., Buyse, M., Sotiriou, C.: Strong Time Dependence of the 76-Gene Prognostic Signature for Node-Negative Breast Cancer Patients in the TRANSBIG Multicenter Independent Validation Series. Clin. Cancer Res. **13**(11), 3207–3214 (2007)
9. Eleuteri, A., Taktak, A.F.G.: Support vector machines for survival regression. In: Biganzoli, E., Vellido, A., Ambrogi, F., Tagliaferri, R. (eds.) CIBB 2011. LNCS, vol. 7548, pp. 176–189. Springer, Heidelberg (2012)
10. Evers, L., Messow, C.M.: Sparse kernel methods for high-dimensional survival data. Bioinformatics **24**(14), 1632–1638 (2008)
11. Harrell, F.E., Califf, R.M., Pryor, D.B., Lee, K.L., Rosati, R.A.: Evaluating the Yield of Medical Tests. J. Am. Med. Assoc. **247**(18), 2543–2546 (1982)
12. Hosmer, D., Lemeshow, S., May, S.: Applied Survival Analysis: Regression Modeling of Time to Event Data. John Wiley & Sons, Inc. (2008)
13. Jensen, P.B., Jensen, L.J., Brunak, S.: Mining electronic health records: towards better research applications and clinical care. Nat. Rev. Genet. **13**(6), 395–405 (2012)
14. Kalbfleisch, J.D., Prentice, R.L.: The Statistical Analysis of Failure Time Data. John Wiley & Sons, Inc. (2002)
15. Kannel, W.B., Feinleib, M., McNamara, P.M., Garrision, R.J., Castelli, W.P.: An Investigation of Coronary Heart Disease in Families: The Framingham Offspring Study. Am. J. Epidemiol. **110**(3), 281–290 (1979)
16. Keerthi, S.S., DeCoste, D.: A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. J. Mach. Learn. Res. **6**, 341–361 (2005)
17. Khan, F.M., Zubek, V.B.: Support vector regression for censored data (SVRc): a novel tool for survival analysis. In: 8th IEEE Int. Conf. on Data Mining, pp. 863–868 (2008)
18. Kimeldorf, G.S., Wahba, G.: A correspondence between bayesian estimation on stochastic processes and smoothing by splines. Ann. Math. Stat. **41**, 495–502 (1970)
19. Lee, C.P., Lin, C.J.: Large-Scale Linear RankSVM. Neural Comput. **26**(4), 781–817 (2014)

20. Mangasarian, O.: A finite newton method for classification. Optimization Methods and Software **17**(5), 913–929 (2002)
21. Ndrepepa, G., Braun, S., Mehilli, J., Birkmeier, K.A., Byrne, R.A., Ott, I., Hösl, K., Schulz, S., Fusaro, M., Pache, J., Hausleiter, J., Laugwitz, K.L., Massberg, S., Seyfarth, M., Schömig, A., Kastrati, A.: Prognostic value of sensitive troponin T in patients with stable and unstable angina and undetectable conventional troponin. Am. Heart J. **161**(1), 68–75 (2011)
22. Shivaswamy, P.K., Chu, W., Jansche, M.: A support vector approach to censored targets. In: 7th IEEE Int. Conf. on Data Mining, pp. 655–660 (2007)
23. Steck, H., Krishnapuram, B., Dehing-oberije, C., Lambin, P., Raykar, V.C.: On ranking in survival analysis: bounds on the concordance index. In: Adv. Neural Inf. Process. Syst., vol. 20, pp. 1209–1216 (2008)
24. Van Belle, V., Pelckmans, K., Suykens, J.A., Van Huffel, S.: Support vector machines for survival analysis. In: Proc. 3rd Int. Conf. Comput. Intell. Med. Healthc, pp. 1–8 (2007)
25. Van Belle, V., Pelckmans, K., Suykens, J.A., Van Huffel, S.: Survival SVM: a practical scalable algorithm. In: Proc. of 16th European Symposium on Artificial Neural Networks, pp. 89–94 (2008)
26. Van Belle, V., Pelckmans, K., Van Huffel, S., Suykens, J.A.K.: Support vector methods for survival analysis: a comparison between ranking and regression approaches. Artif. Intell. Med. **53**(2), 107–118 (2011)