Boosting the First-Hitting-Time Regression Model

Vegard Stikbakke

November 15, 2018

Abstract

Empty.

Acknowledgements

Empty.

Contents

Abstract **i**

Acknowledgements **iii**

Contents **v**

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

2 First hitting time regression models in survival analysis **3**
2.1 Survival analysis and time-to-event models 3
2.2 The First Hitting Time (FHT) Model 5
2.3 First hitting time regression based on underlying Wiener process 6
2.4 Likelihood 7
2.5 Simulation of survival data 8

3 Statistical boosting **11**
3.1 Statistical learning theory 11
3.2 The history of boosting 12
3.3 Statistical boosting 12
3.4 Finding a solution 13
3.5 Gradient descent 13
3.6 Gradient boosting: Functional gradient descent 14
3.7 L2Boost 14
3.8 Component-wise gradient boosting 16
3.9 Multidimensional boosting: Component-wise boosting of a multivariate loss function 16
3.10 The importance of stopping early 17
3.11 Selecting m_{stop} 17

4 First hitting time boost **21**
4.1 Algorithm 21

Appendices **23**

A Appendix 1: Differentiating the IG FHT **25**

Bibliography **29**

List of Figures

List of Tables

Chapter 1

Introduction

sec:intro

In this thesis, we work with boosting for regression in the first hitting time model. First hitting time is a model in survival analysis which serves as an alternative to the proportional hazards model, typically known as Cox regression. Developments in FHT regression are relatively recent, and there has to our knowledge been no attempt at tackling it in the high-dimensional case, in which boosting is an appropriate choice of method.

Chapter 2

First hitting time regression models in survival analysis

2.1 Survival analysis and time-to-event models

sec:survival

Lifetimes and time-to-event data are of interest in many applications. Oncologists, doctors who study cancer, are interested in how quickly patients die after cancer has been discovered. Sociologists might be interested in the duration of marriages before divorce. We say that a lifetime T ends when an event occurs. In the previous examples, the events in question are death and divorce, say. We are usually interested in making inference about this lifetime, and in particular what factors it depends upon. In biomedical fields, this is known as survival analysis, while in engineering it is called reliability analysis. These are much studied fields. The main part of our thesis is applicable in both areas. In the former case, we may consider the time before a component of a system breaks and must be replaced. Let the lifetime or time-to-event T be a continuous non-negative random variable following a cumulative distribution function $F(t)$, such that

$$F(t) = \Pr(T < t)$$

is the probability of the event having happened before time t. We define the survival function $S(t)$ to be the converse, namely

$$S(t) = 1 - F(t).$$

Hence, $S(t)$ denotes the probability that the event has not yet happened at time t. If the cumulative distribution function is differentiable, we define the probability density function of T to be $f(t)$. Another much studied property of lifetime distributions is the hazard function $h(t)$. Somewhat informally, it denotes the probability of the event happening at some time t, assuming it has not happened yet. More formally, it is the limit of the conditional probability that the event will occur in a small interval $[t, t + \Delta t)$, conditional on the event not having happened yet,

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr(t \leq T < t + \Delta t | T \geq t)}{\Delta t} = \frac{f(t)}{S(t)}.$$

Censored data

In theory, a lifetime will always end. In the real world, however, we are constrained with finite time. Thus, when we observe lifetime data, it is not necessarily the case that the lifetime has ended yet. For example, some cancer patients survive, and die of old age. Similarly, some marriages do not end in divorce. In these cases, it is likely that the lifetime has not ended at the time we stop observing data. Here we cannot say anything about the lifetime itself. We can only say that this lifetime lasted at least until now. We call these observations censored observations. Specifically, these are right-censored observations, since they are censored at the right end of the time scale.

sec:surv-data

Data structures

Assume we observe t_i, $i = 1, \ldots, n$ independent and identically distributed (*iid*) observations from a random variable with density distribution f. For a single individual event i, we might observe the following. The time t_i of the observation. Covariates $\mathbf{x}_i$ describing the individual. An indicator δ_i of whether the individual event has occurred ($\delta_i = 1$) or not ($\delta_i = 0$). The latter events, corresponding to $\delta_i = 0$, are censored. We are interested in setting up the likelihood, where we incorporate the covariates $\mathbf{x}_i$ into a parameter $\boldsymbol{\theta}$. If the event has occurred, the indicator δ_i is 1. We can then use the information about the lifetime distribution, such that the single individual i contributes

$$f(t_i|\mathbf{x}_i) \tag{2.1}$$

{eq:f}

to the likelihood. If the event has not (yet) occurred, the observation is censored, and δ_i is 0. Since we do not have the actual lifetime, we cannot use the lifetime distribution. We must use the survival distribution. As such, this observation contributes

$$S(t_i|\mathbf{x}_i) \tag{2.2}$$

{eq:S}

to the likelihood. Obviously, since an observation can only be either censored or not censored at the same time, δ_i is either 0 or 1. If the event has occurred, δ_i is 1, and then $1 - \delta_i$ is 0. Similarly, if the event has not occurred and the event is censored, then δ_i is 0, and then $1 - \delta_i$ is 1. This allows us to take the product of (2.1) and (2.2) where we take these to the power of δ_i and $1 - \delta_i$, respectively, so that a single observation makes a contribution of

$$f(t_i|\mathbf{x}_i)^{\delta_i} S(t_i|\mathbf{x}_i)^{1-\delta_i}$$

to the likelihood. Since we assumed the observations were independent, the likelihood of the observed sample as a whole is the product of the single likelihoods. The likelihood becomes

$$L(\boldsymbol{\theta}|\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(n)}) = \prod_{i=1}^{n} f(t_i|\mathbf{x}_i, \boldsymbol{\theta})^{\delta_i} S(t_i|\mathbf{x}_i, \boldsymbol{\theta})^{1-\delta_i}. \tag{2.3}$$

{eq:surv-lik}

Proportional hazards

The most used method for doing regression on survival data is the Cox proportional hazards (PH) regression. It is based on an assumption that is often

called the PH property or the PH assumption, namely that

$$h(t|x) = h_0(t)g(\mathbf{x}), \tag{2.4}$$

where $h_0(t)$ is a baseline hazard function, which is common for all individuals. This means that at any two time points t_1 and t_2, the ratio between the hazard functions of any two $\mathbf{x}_1$ and $\mathbf{x}_2$ will be the same:

$$\frac{h(t_1|x_1)}{h(t_1|x_2)} = \frac{h(t_2|x_1)}{h(t_2|x_2)} \tag{2.5}$$

This is a strong assumption to make, and it will not always be the case in practice (Lee and Whitmore, 2010). Therefore there is a need for more flexible methods in survival analysis, where this assumption is not necessary.

2.2 The First Hitting Time (FHT) Model

sec:fht

Time-to-event data analysis in biomedical sciences is dominated by Cox regression, which is a semiparametric proportional hazards model, and which directly estimates the hazard rate (Stogiannis and Caroni, 2013). A class of parametric models which has got increasingly more attention recently is the first hitting time (FHT) model, originally developed by Whitmore in 1986 (Whitmore, 1986; Lee and Whitmore, 2006). The FHT model has been applied successfully to different kinds of data, especially in biomedial sciences. Examples include modelling lung cancer risk in railroad workers (Lee et al., 2004), ... In an FHT model, the health of an individual is modelled as a stochastic process, which, when it reaches some threshold (or, more generally, an absorbing state), triggers the event, at which point the lifetime ends. The time-to-event, or lifetime, becomes the time it takes for the process to reach this state. This is an attractive model because it models the process instead of the hazard rate (Aalen and Gjessing, 2001). It is also conceptually appealing, because it makes sense to imagine that there is some process governing when events happen, such that for two living individuals, they might have different distances, so to speak, away from death. We now descibe the key components in the FHT model. There is a parent stochastic process $\{Y(t)\}$, $Y \in \mathcal{Y}$, time t non-negative, $t \in \mathcal{T}$. The process has initial value $Y(0) = y_0$. There is a boundary set $\mathcal{B} \subset \mathcal{T}$, which is at times referred to as a boundary, barrier, or threshold, all of which are synonymous. The preferred term varies with which interpretation we want to use and what connotations we want to evoke. The choice of process is flexible. It might have continuous or discrete sample paths. We define the first hitting time S to be the first time t that the process Y reaches the absorbing state $B \in \mathcal{B}$,

$$S = \inf\{t \colon Y(t) \in \mathcal{B}\}. \tag{2.6}$$

Note that by definition $y_0 \notin \mathcal{B}$, since the event has not yet happened at time $t = 0$. Note also that it is quite possible that the process does not reach an absorbing state, and so that $P(S < \infty) < 1$. The FHT model does not require the PH assumption, and is hence more flexible. In fact, the PH model may be obtained by constructing the first hitting time model in a specific way (Lee and Whitmore, 2010). Different choices for the process Y lead to different kinds of distributions for the first hitting time. We now look at some common choices for the process.

sec:wiener

Wiener process

The Wiener process, also known as the standard Brownian motion process, is a process which is continuous in time and space. It is so far the most commonly used process in FHT literature. The Wiener process is a fairly simple process: It has three parameters, the drift μ, the variance σ^2, and the initial value $Y(0) = y_0$. It has independent increments, such that $Y(t_2) - Y(t_1)$ and $Y(t_4) - Y(t_3)$ are independent for any disjoint intervals (t_1, t_2) and (t_3, t_4). Each increment is normally distributed and with both the mean and the standard deviation proportional to the length of the interval, i.e., for any interval (t_1, t_2),

$$Y(t_2) - Y(t_1) \sim N(\mu(t_2 - t_1), \sigma^2(t_2 - t_1)). \tag{2.7}$$

If we consider Y to be a Wiener process in the FHT model, we will typically choose the boundary $\mathcal{B}$ to be $\mathcal{B} = (-\infty, 0]$, i.e., the event is triggered when $Y \leq 0$. Accordingly, we then also assume that the initial level y_0 is positive. This is a very conceptually appealing model, because it assumes that individuals might have different initial levels, and that also the drift might be different between individuals. It is also attractive because it has closed-form probability and cumulative density functions, and its likelihood is computationally simple. There are no restrictions on the movements of the process, meaning, it is non-monotonic. If we do want a monotonic restriction on the movement of the process, we may use a gamma process.

Other processes

The gamma process is suitable for modelling a process which we would require to be monotonic, typically a physical degradation, i.e. where the damage cannot mend itself, unlike a patient's health. The first hitting time that arises from the gamma process is an inverse gamma first hitting time distribution (Lee and Whitmore, 2006). Other choices of processes include Markov chain state models, the Bernoulli process, and the Ornstein-Uhlenbeck process. For a complete review, see Lee and Whitmore (2006). Due to its simplicity and flexibility, we will in thesis focus on the Wiener process as the choice of process in the FHT model. However, large parts of our results can easily be extended for other processes. For brevity, we will in the sequel say "the FHT" when we in fact mean the FHT with the Wiener process.

2.3 First hitting time regression based on underlying Wiener process

It can be shown that the first hitting time of the Wiener process follows an inverse Gaussian distribution (Chhikara, 1988),

$$f(t|y_0, \mu, \sigma^2) = \frac{y_0}{\sqrt{2\pi\sigma^2 t^3}} \exp\left[-\frac{(y_0 + \mu t)^2}{2\sigma^2 t}\right]. \tag{2.8}$$

{eq:fht-ig}

TO DO!

See Appendix for the mathematical derivation. If the drift μ is positive, then it is not certain that the process will reach 0, so in this case it is an improper pdf. In most cases, y is not measured directly. If that is the case, then the scale of y is arbitrary. Thus, we may fix one parameter in the distribution. As

per convention we choose to set the variance to unity, i.e., $\sigma^2 = 1$ (Lee and Whitmore, 2006; Caroni, 2017). While μ and y_0 have simple interpretations in terms of the underlying process, they do not in terms of the lifetime distribution. The mean lifetime is $\frac{y_0}{|\mu|}$, and its variance is $\frac{y_0}{|\mu|^3}$ (Caroni, 2017). This

The cumulative distribution function of the FHT is (Xiao et al., 2015)

$$F(t|\mu, \sigma^2, y_0) = \Phi\left[-\frac{(\mu t + y_0)}{\sqrt{\sigma^2 t}}\right] + \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left[\frac{\mu t - y_0}{\sqrt{\sigma^2 t}}\right], \tag{2.9}$$

{eq:cumulative}

where $\Phi(x)$ is the cumulative distribution function of the standard normal, i.e.,

$$\Phi(x) = \int_{-\infty}^{x} \exp(-y^2/2)/\sqrt{2\pi}\ \mathrm{d}y. \tag{2.10}$$

The survival function $S(t)$ is

$$\begin{aligned} S(t) &= 1 - \Phi\left[-\frac{(\mu t + y_0)}{\sqrt{\sigma^2 t}}\right] - \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left[\frac{\mu t - y_0}{\sqrt{\sigma^2 t}}\right] \\ &= \Phi\left[\frac{\mu t + y_0}{\sqrt{\sigma^2 t}}\right] - \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left[\frac{\mu t - y_0}{\sqrt{\sigma^2 t}}\right]. \end{aligned} \tag{2.11}$$

{eq:survival}

In the last step we use the fact that $1 - \Phi(-x) = \Phi(x)$, since the standard normal distribution is symmetric around 0.

Regression

We may introduce effects from covariates by allowing μ and y_0 to depend on covariates $\mathbf{x}$ and $\mathbf{z}$. A simple and much used model is to use the identity and the logarithm link function for the drift μ and the initial level y_0, respectively.

$$\begin{aligned} \mu &= \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x} \\ \ln y_0 &= \boldsymbol{\gamma}^{\mathrm{T}}\mathbf{z} \end{aligned} \tag{2.12}$$

{eq:coeffs}

$\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are vectors of regression coefficients. Note that we may let $\mathbf{x}$ and $\mathbf{z}$ share none, some, or all elements.

2.4 Likelihood

sec:lik

In formula (2.3), we reported the likelihood of lifetime regression models in its most general formulation. For an inverse Gaussian FHT this then becomes (inserting (2.8) and (2.11) into (2.3))

$$\begin{aligned} L(\boldsymbol{\theta}) = &\left(\frac{y_0}{\sqrt{2\pi\sigma^2 t^3}}\exp\left[-\frac{(y_0 + \mu t)^2}{2\sigma^2 t}\right]\right)^{\delta_i} \\ &\times \left[\Phi\left(\frac{\mu_t + y_0}{\sqrt{\sigma^2 t}}\right) - \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t - y_0}{\sqrt{\sigma^2 t}}\right)\right]^{1-\delta_i} \end{aligned} \tag{2.13}$$

{eq:fht-lik}

We can now substitute the covariates in (2.12) into this.

Optimization

Until now, mainly numerical maximum likelihood methods have been used to find optimal parameters, via direct maximization of the likelihood. Finding a closed-form solution for the maximum likelihood is not possible. It is only feasible to do numerical optimization of the maximum likelihood in the low-dimensional case, since it will optimize the entire parameter space at once. Therefore it is necessary to develop methods which can deal with high-dimensional cases. That is what we intend to do in the main part of the thesis. For our purposes, we need to differentiate the logarithm with respect to the parameters μ and y_0. Since the logarithm is monotone, it preserves optimality, and hence we can take the logarithm of (2.13), and we get

$$\begin{aligned} l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \delta_i \Bigg(& \ln y_0 - \frac{1}{2}\ln\left(2\pi\sigma^2 t_i^3\right) - \frac{(y_0+\mu t_i)^2}{2\sigma^2 t_i} \Bigg) \\ & + (1-\delta_i)\ln\Bigg(\Phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) - \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\Bigg) \end{aligned} \tag{2.14}$$

{eq:loglik}

See the appendix A for the full derivation.

Parametrizations of inverse Gaussian

The standard parametrization of the Inverse Gaussian is

$$f(x;\mu,\lambda) = \frac{\sqrt{\lambda}}{\sqrt{2\pi x^3}} \exp\left(-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right)$$

From Chhikara (1988) we know that if the underlying Brownian motion with initial level $y_0 > 0$ has negative drift ν, i.e., drifts toward the barrier 0, then it is Inverse Gaussian with the above parameterization, with

$$\mu = \frac{y_0}{-\nu}$$
$$\lambda = \frac{y_0^2}{\sigma^2}$$

2.5 Simulation of survival data

We wish to simulate survival times $t_i, i = 1, \ldots, N$ with censoring. We first draw survival times $\tilde{t}_i$ from some survival time distribution $f(\cdot)$. If this distribution has a closed form probability distribution function, we can draw from it directly. To censor the data, we draw censoring times $W_i \sim f(\cdot), i = 1, \ldots, N$, from a more right-tailed distribution, meaning we want to get many, but not all, W_i's to be larger than the $\tilde{t}_i$'s. We let the observed survival times then be $t_i = \min(\tilde{t}_i, W_i)$. The corresponding observed indicator, δ_i, is then set equal to 1 if the actual survival time was observed, i.e., if $t_i < W_i$. We end up with a set of N tuples $(t_i, \delta_i), i = 1, \ldots, N$. Note that this scheme incorporates independent censoring: The censoring time is independent of the survival times. This does not pose a problem. Summary of the procedure:

algo:sim

Algorithm 1 Generate data

1. Make design matrices $\mathbf{X}$, $\mathbf{Z}$.
2. Set $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.
3. Link covariates and parameters using link functions

$$\ln y_0 = \boldsymbol{\beta}^T \mathbf{X}$$
$$\mu = \boldsymbol{\gamma}^T \mathbf{Z}.$$

4. Draw N survival times $(t_i)_{i=1}^N$ from $\mathrm{IG}(\mu, y_0)$.
5. Draw censoring times W from some distribution.
6. Right censor data by choosing $\widetilde{t}_i = \min(t_i, W)$. The indicator on whether observation i was observed or not is then $\delta_i = I(\widetilde{t}_i = t_i)$.
7. The simulated data set is (t_i, δ_i).

Chapter 3

Statistical boosting

3.1 Statistical learning theory

sec:learning-theory

Assume we have a joint distribution $(\mathbf{X}, Y)$, $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$, and $Y = F(\mathbf{X}) + \varepsilon$, $F(\mathbf{x}) = \mathrm{E}(Y|\mathbf{X} = \mathbf{x})$ We wish to estimate the underlying $F(\mathbf{X})$. For an estimate $\hat{F}(\cdot)$, we measure the loss, or the difference, with a loss function

$$\mathrm{L}(Y, \hat{F}(\mathbf{X})).$$

A common loss function for regression is the squared loss, also known as the L_2 norm,

$$\mathrm{L}(Y, \hat{F}(\mathbf{X})) = (Y - \hat{F}(\mathbf{X}))^2.$$

For a $\hat{F}(X)$, we wish to estimate the expected loss, also known as the generalization or test error,

$$\mathrm{Err}_\tau = \mathrm{E}[\mathrm{L}(Y, \hat{F}(\mathbf{X}))|\tau],$$

where (X, Y) is drawn randomly from their joint distribution and the training set τ is held fixed. It is infeasible to do effectively in practice and hence we must instead estimate the expected prediction error,

$$\mathrm{Err} = \mathrm{E}[\mathrm{Err}_\tau] = \mathrm{E}_\tau \left([\mathrm{L}(Y, \hat{F}(\mathbf{X}))|\tau] \right), \tag{3.1}$$

{eq:err}

i.e., average over many different test sets. In practice, we observe a sample $(\mathbf{x}_i, y_i)_{i=1}^N$. For this sample, we can calculate the training error,

$$\overline{\mathrm{err}}(F) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{L}(y_i, \hat{F}(\mathbf{x}_i)), \tag{3.2}$$

{eq:empirical-risk}

also known as the empirical risk. To estimate err (3.1), one can do two things. First, if the observed sample is large enough, one can choose a portion of this, say 20%, to be used as a hold-out test set. We then train/fit/estimate based on the other 80%, and estimate Err by

$$\hat{\mathrm{Err}} = \frac{1}{M} \sum_{i=1}^{M} L(y_i, \hat{F}(\mathbf{x}_i)),$$

where (x_i, y_i) here are from the test set.

3.2 The history of boosting

history

Boosting is one of the most promising methodological approaches for data analysis developed in the last two decades (Mayr et al., 2014). Boosting originated in the fields of computational learning theory and machine learning. Kearns and Valiant (1989), working on computational learning theory, posed a question: Could any weak learner be transformed to become a strong learner? A weak learner, sometimes also simple or base learner, means one which has a low signal-to-noise ratio, and which in general performs poorly. For classification purposes it is easy to give a good example: A weak learner is one which performs only slightly better than random uniform chance. Freund and Schapire (1996) then invented the AdaBoost algorithm for constructing a binary classifier. It was evidence that the answer to the original question was positive. The AdaBoost algorithm performs iterative reweighting of original observations. For each iteration, it gives more weight to misclassified observations, and then trains a new weak learner based on all weighted observations. It then adds the new weak learner to the final classifier. The resulting AdaBoost classifier is then a linear combination of these weak classifiers, i.e., a weighted majority vote. In its original formulation, the AdaBoost classifier does not have interpretable coefficients, and as such it is a so-called black-box algorithm. In statistics, however, we are interested in models which are interpretable.

3.3 Statistical boosting

sec:sboost

While AdaBoost often has good predictive performance, in its original formulation, it is a black box algorithm. This means that we are unable to infer anything about the effect of different covariates. In statistics, we often want to estimate the relation between observed predictor variables and the expectation of the response,

$$\mathrm{E}(Y|\mathbf{X} = \mathbf{x}) = F(\mathbf{x}). \tag{3.3}$$

{eq:exp-f}

In addition to using boosting for classification, like in the original AdaBoost, we would also like to use it in more general settings, and we therefore extend our discussion to a more general regression scheme where the outcome variable Y can be continuous. We are interested in interpreting how the different covariates of $\mathbf{X}$ affect $\mathrm{E}(\cdot)$. A choice for $F(\mathbf{X})$ which is amenable to such interpretation is the generalized additive model (GAM),

$$F(\mathbf{x}) = \alpha + \sum_{j=1}^{p} f_j(x_j), \tag{3.4}$$

{eq:gam}

where $\alpha \in \mathbb{R}$ is an offset and x_j is the j-th component of $\mathbf{x}$. $F(\mathbf{x})$ is a sum of component-wise functions f_j, and as such a GAM contains interpretable additive predictors. Friedman et al. (2000) showed that AdaBoost fits a GAM with a forward stagewise algorithm, for a particular exponential loss function. This provided a way of viewing boosting through a statistical lens. Friedman (2001) further investigated the topic, providing further insight into boosting. However, we must first discuss how we find an approximate solution for $F(\mathbf{X})$ in (3.3).

3.4 Finding a solution

We wish to find $F(\mathbf{X})$ in (3.3), so we are interested in solving

$$\underset{F}{\operatorname{argmin}} \operatorname{E}\left[\operatorname{L}(Y, F(\mathbf{X}))\right],$$

where L is an appropriate loss function. But as discussed in chapter 3.1, we have in practice observed a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$, drawn from the joint distribution of $(\mathbf{X}, Y)$. Therefore we substitute the expected loss with the expected risk (3.2), and so we want a solution to

$$\underset{F}{\operatorname{argmin}} \overline{\operatorname{err}}(F). \tag{3.5}$$

{eq:argmin-risk}

Finding F is not easy. We will now discuss a general optimization algorithm used in boosting.

3.5 Gradient descent

Gradient descent is an optimization algorithm for minimizing a differentiable multivariate function F. The motivation behind the gradient descent algorithm is that in a small interval around a point $\mathbf{x}$, F is decreasing in the direction of the negative gradient at $\mathbf{x}$. Therefore, by moving slightly in that direction, F will decrease. Indeed, with a sufficiently small step length, gradient descent will always converge, albeit to a local minimum. For a schematic overview of the algorithm, see Algorithm 2. Back to our goal of finding an F to minimize

algo:grad-desc

Algorithm 2 Gradient descent

1. Start with an initial guess $\mathbf{x}_0$, e.g. $\mathbf{x}_0 = \mathbf{0}$. Let $m = 1$.
2. Calculate the direction to step in, $\mathbf{g}_{m-1} = -\nabla F(\mathbf{x}_{m-1})$.
3. Let $\mathbf{h}_m = \nu \mathbf{g}_{m-1}$, where ν is the best step length, found by
$$\nu = \underset{\nu}{\operatorname{argmin}} \mathbf{x}_{m-1} + \nu \mathbf{g}_{m-1}$$
4. Let $\mathbf{x}_m = \mathbf{x}_{m-1} + \mathbf{h}_{m-1}$
5. Increase m, and go to step 2. Repeat until $m = M$.
6. The resulting final guess is $\mathbf{x}_M = \mathbf{x}_0 + \sum_{m=1}^M \mathbf{h}_m(\mathbf{x}_m)$

(3.5). Often we choose a parameterized model $F(\mathbf{X}; \boldsymbol{\beta})$. Finding the optimal $\boldsymbol{\beta}$ analytically might be infeasible. The gradient descent algorithm can then be used. In this case, we fix $\mathbf{X}$ and let $F(\mathbf{X})$ in the algorithm be $F(\boldsymbol{\beta}; \mathbf{X})$. Thus we use gradient descent to find an optimal $\boldsymbol{\beta}$. We would then say we are doing gradient descent in parameter space. We are now ready to reveal Friedman's useful insight.

3.6 Gradient boosting: Functional gradient descent

There is another possible way to use gradient descent, and that is the important insight by Friedman in 2001 (Friedman, 2001). He showed that boosting can be viewed as an optimization procedure in functional space. Briefly, we first describe a naive way of doing this. Consider the function value at each $\mathbf{x}$ directly as a parameter, and use gradient descent directly on these parameters. However, this does not generalize to unobserved values $\mathbf{X}$, and we are after all interested in the population minimizer of (3.3). We can instead assume a parameterized form for F, e.g.,

$$F(\mathbf{X}; \{\boldsymbol{\beta}\}_{m=1}^{M}) = \sum_{m=1}^{M} \nu H(\mathbf{X}; \boldsymbol{\beta}_m), \tag{3.6}$$

{eq:gradboost}

where $H(\mathbf{X}; \boldsymbol{\beta})$ is also a function on the GAM form (3.4), but typically simpler, i.e., a base learner as discussed previously. We would like to minimize a data based estimate of the loss, i.e. the empirical risk, and so would choose $\{\boldsymbol{\beta}_m\}$ as the minimizers of

$$\underset{\{\boldsymbol{\beta}_m\}_{m=1}^{M}}{\operatorname{argmin}} \overline{\operatorname{err}}(H(\mathbf{x}; \{\boldsymbol{\beta}_m\})).$$

However, estimating these simultaneously may be infeasible. We can then use a greedy stagewise approach, where we at each step m choose the $\boldsymbol{\beta}_m$ which gives the best improvement, while not changing any of the previous $\{\boldsymbol{\beta}\}_{k=1}^{m-1}$. Hence at each step m the current solution is

$$F_m = F_{m-1} + \nu H(\mathbf{x}; \boldsymbol{\beta}_m),$$

where the parameters $\boldsymbol{\beta}_m$ are those in H minimizing the empirical risk when added to the fixed part F_{m-1}:

$$\boldsymbol{\beta}_m = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \overline{\operatorname{err}}(H(\mathbf{x}; \boldsymbol{\beta}_k) + H(\mathbf{x}; \boldsymbol{\beta})).$$

The final model is then the sum of these terms, like in (3.6). To find $\boldsymbol{\beta}_m$ in each step here, we use gradient descent. We have outlined a generic functional gradient descent algorithm. For a schematic overview of this, see Algorithm 3. Note that while we call this functional gradient descent (FGD), this is exactly the gradient boosting algorithm.

3.7 L2Boost

Based on Friedman (2001)'s results, Bühlmann and Yu (2003) developed the L2Boost algorithm. It is a special case of the generic functional gradient descent (FGD) algorithm, where we choose the squared error loss to be the loss function,

$$\mathrm{L}(y, F(\mathbf{x})) = \frac{1}{2} \left(y - F(\mathbf{x})\right)^2.$$

The negative gradient vector of the loss then becomes the residual vector,

$$\frac{\partial \mathrm{L}(y, F(\mathbf{x}))}{\partial x_i} = (y - F(x_i)), \quad i = 1, \ldots, n,$$

algo:fgd

Algorithm 3 Functional gradient descent

1. Initialize $F_0(\mathbf{x})$, e.g., by setting it to zero for all components. Select a base learner H.
2. Compute the negative gradient vector,
$$U_i = -\frac{\partial \mathrm{L}(y_i, F_{m-1}(\cdot))}{\partial F_{m-1}(\cdot)}, \quad i = 1, \ldots, N.$$
3. Estimate $\hat{H}_m$ by fitting $(\mathbf{X}_i, U_i)$ using the base learner H (like in the previous algorithm):
$$\boldsymbol{\beta}_m = \operatorname*{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N} \mathrm{L}(u_i, H(\mathbf{x}_i; \boldsymbol{\beta}))$$
4. Update $F_m(\cdot) = F_{m-1}(\cdot) + \nu H(\cdot; \boldsymbol{\beta}_m)$.
5. Repeat steps from 2 until $m = M$.

and hence the boosting steps become repeated refitting of residuals (Friedman, 2001; Bühlmann and Yu, 2003). With $\nu = 1$ and $M = 2$, this had been proposed already by (Tukey, 1977), who called it "twicing". See Algorithm 4 for an overview.

algo:l2

Algorithm 4 L2Boost

1. Initialize $F_0(\mathbf{x})$, e.g., by setting it to zero for all components. Select a base learner H, such as ordinary least squares, stumps, etc.
2. Compute the residuals
$$U_i = (y_i, F_{m-1}(x_i)), \quad i = 1, \ldots, n$$
3. Estimate $\hat{H}_m$ by fitting $(\mathbf{X}_i, U_i)_{i=1}^N$ using the base learner H:
$$\boldsymbol{\beta}_m = \operatorname*{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N} \mathrm{L}(u_i, H(\mathbf{x}_i; \boldsymbol{\beta}))$$
Note that $\hat{H}(\cdot; \boldsymbol{\beta}_m)$ is then an estimate of the negative gradient vector.
4. Update $F_m(\cdot) = F_{m-1}(\cdot) + \nu H(\cdot; \boldsymbol{\beta}_m)$.
5. Repeat steps from 2 until $m = M$..

They also prove some important theoretical results for L2Boost.

3.8 Component-wise gradient boosting

In high-dimensional settings, it might often be infeasible, if not impossible, to use a base learner H which incorporates all p dimensions. Indeed, using least squares base learners, it is impossible, since the matrix which must be inverted is singular when $p > N$. Component-wise gradient boosting is a technique/algorithm which does work in these settings. First developed by Bühlmann and Yu (2003), and it has further been refined and explored, see e.g. Bühlmann (2006). The idea of the algorithm is to select a set of base learners, the most important property of which being that they are univariate: Each base learner is only a function of one component x_j of the data $\mathbf{X}$, i.e.,

$$h_j(\mathbf{x}) = h_j(x_j).$$

In each iteration, we fit the learners separately, and choose only the one which gives the best improvement to be added in the final model. The resulting model $F_m(\cdot)$ is then a sum of componentwise effects,

$$F_m(\mathbf{X}) = \sum_{j=1}^{p} f_j(x_j),$$

where

$$f_j(x_j) = \sum_{m=1}^{M} \mathbb{1}_{mj} h_j(x_j; \boldsymbol{\beta}_m),$$

where $\mathbb{1}_{mj}$ is an indicator function which is 1 if component j was selected at iteration m and 0 if not. Hence this model is a GAM (3.4). Crucially, if we stop sufficiently early, we will typically perform variable selection. It is likely that some base learners have never been added to the final model, and as such those components in $\mathbf{X}$ are not added. For a schematic overview of the algorithm, see Algorithm 5. In fact, Buhlmann believes that it is mainly in the case of

algo:component-gradboost

Algorithm 5 Component-wise gradient boosting

1. Start with an initial guess, e.g. $F_0 = \mathbf{0}$.
   Specify a set of base learners $h_1(\cdot), \ldots, h_p(\cdot)$.
2. Compute the negative gradient vector $\mathbf{U}$.
3. Fit $(\mathbf{X}_i, U_i)_{i=1}^N$ separately to every base learner to get $\hat{h}_1(x_1), \ldots, \hat{h}_p(x_p)$.
4. Select the component k which best fits the negative gradient vector.
$$k = \underset{j \in [1,p]}{\operatorname{argmin}} \sum_{i=1}^{N} (u_i - \hat{h}_j(\mathbf{x}_i))^2$$
5. Update $F_m(\cdot) = F_{m-1}(\cdot) + \nu h_k(x_k)$

high-dimensional predictors that boosting has a substantial advantage over classical approaches (Bühlmann, 2006).

3.9 Multidimensional boosting: Component-wise boosting of a multivariate loss function

The above methods consider a loss function depending on one parameter: $L(\beta)$. In the boosting steps, one uses a gradient descent step with the loss function differentiated with respect to this one parameter. There are however many models which use loss functions of several variables. Schmid et al. (2010) extend the gradient boosting algorithm (Friedman, 2001) to such a setting. In this method, we take the partial derivative of the multivariate loss function with respect to each variable. They propose doing a boosting step in each dimension at a time, and repeat this for all dimensions. We also find an m_{stop} for each dimension. The algorithm proposed in Schmid et al. (2010) does an additional cycle through of nuisance parameters after having cycled through all dimensions, in each boosting step. The algorithm is as follows.

3.10 The importance of stopping early

The number of iterations in the boosting procedure, M, is a tuning parameter. It acts as a regularizer.

3.11 Selecting m_{stop}

The crucial tuning parameter in boosting is the number of iterations, m_{stop}. Stopping early enough performs variable selection and shrinks the parameter estimates toward zero. Left on its own, the parameters in boosting will converge towards the maximum likelihood parameters , i.e., maximizing the in-sample error. We are, on the other hand, after all interested in minimizing out-of-sample prediction error (PE). The prediction error for a given data set is a function of the boosting iteration m. What we want is therefore a good method for approximating $\text{PE}(m)$. This can be done in a number of ways. Many authors state that the algorithm should be stopped early, but do not go further into the details here. Common model selection criteria such as the Akaike Information Criteria (AIC) may be used, however the AIC is dependant on estimates of the model's degrees of freedom. Methods by Chang et al. (2010) try this. This is problematic for several reasons. For L_2Boost, Bühlmann and Hothorn (2007) suggest that $\text{df}(m) = \text{trace}(B_m)$ is a good approximation. Here B_m is the hat matrix resulting from the boosting algorithm. This was, however, shown by Hastie (2007) to always underestimate the actual degrees of freedom. Mayr et al. (2012) propose a sequential stopping rule using subsampling etc. We argue instead that cross-validation, a very common method for selection of tuning parameters in statistics, is a good method to use. It is flexible and easy to implement. It is somewhat computationally demanding, requiring several full runs of the boosting algorithm.

SOURCE?

K-fold cross-validation

K-fold cross-validation (Lachenbruch and Mickey, 1968), or simply cross-validation, is a general method commonly used for selection of penalty or tuning parameters. We will use it to approximate the prediction error. In

algo:multidim-boost

Algorithm 6 K-dimensional component-wise gradient boosting

1. Initialize the n-dimensional vectors $f_1^{[0]}, \ldots, f_K^{[0]}$, with offset values, e.g. with $f_1^{[0]} = \mathbf{0}, \ldots, f_K^{[0]} = \mathbf{0}$. Alternatively, one can use the maximum likelihood estimates as offset values.

2. For each component $k = 1, \ldots, K$, specify a base learner to use for each of the p components. The base learner takes one input variable and has one output variable. Examples include.

3. Set $m = 0$.

4. Increase m by 1.

   a) Set $k = 0$.

   b) Increase k by 1. If $m > m_{\text{stop},k}$, proceed to step 4 f). If not, compute the negative partial derivative $-\frac{\partial \rho}{\partial f_k}$ and evaluate at $\hat{f}^{[m-1]}(X_i) = \left(\hat{f}_1^{[m-1]}(X_i), \ldots, \hat{f}_K^{[m-1]}(X_i)\right), i = 1, \ldots, n$. This yields the negative gradient vector $U_k^{[m-1]} = \left(U_{i,k}^{[m-1]}\right)_{i=1,\ldots,n}$ $:= \left(-\frac{\partial}{\partial f_k}\rho\left(Y_i, \hat{f}^{[m-1]}(X_i)\right)\right)_{i=1,\ldots,n}$.

   c) Fit the negative gradient vector $U_k^{[m-1]}$ to each of the p components of $\mathbf{X}$ separately (i.e. to each predictor variable) using the base learners specified in step 2. This yields p vectors of predicted values, where each vector is an estimate of the negative gradient vector $U_k^{[m-1]}$.

   d) Select the component of $\mathbf{X}$ which best fits $U_k[m-1]$ best according to a pre-specified goodness-of-fit criterion. For continuous variables, the R^2 measure should (?) be used. Set $\hat{U}_k^{[m-1]}$ equal to the fitted values of the corresponding best model fitted in the previous step.

   e) Update $\hat{f}_k[m-1] \leftarrow \hat{f}_k^{[m-1]} + \nu \hat{U}_k^{[m-1]}$, where ν is a real-valued step-length factor.

   f) For $k = 2, \ldots, K$, repeat steps. Finally, update $\hat{f}^{[m]} \leftarrow \hat{f}^{[m-1]}$.

5. Iterate step 4 until $m > \max(m_{\text{stop},k})$ for all $k \in \{1, \ldots, K\}$.

cross-validation, the data is split randomly into K rougly equally sized folds. For a given fold k, all folds except k act as the training data in estimating the model. We often say that the k'th fold is left out. The resulting model is then evaluated on the unseen data, namely fold k. This procedure is repeated for all $k = 1, \ldots, K$. An estimate for the prediction error is obtained by summing over the test errors from evaluting the left-out fold. Let $\kappa(k)$ be the set of indices for fold k. The cross-validated estimate for a given m then becomes

$$\text{CV}(m) = \sum_{k=1}^{K} \sum_{i \in \kappa(k)} \text{L}\left((t_i, \delta_i), \theta_m; \mathbf{x}_i\right). \tag{3.7}$$

For each m, we calculate the estimate of the cross-validated prediction error $\text{CV}(m)$. We choose m_{stop} to be the minimizer of this error,

$$m_{\text{stop}} = \underset{m}{\text{argmin}}\, \text{CV}(m). \tag{3.8}$$

Typical values for K are 5 or 10, but in theory one can choose any number. The extreme case is $K = N$, called leave one out cross-validation, where all but one observation is used for training and one evaluates the model on the observation that was left out. In this case, the outcome is deterministic, since there is no randomness when dividing into folds.

Stratified cross-validation

When dividing an already small number of survival data observations into K folds, we might risk getting folds without any observed deaths, or in any case, very few. In stratified cross validation, we do not divide the folds entirely at random, but rather, try to divide the data such that there is an equal amount of censored data in each fold. As before, let $\kappa(k)$ be the set of indices for fold k. Divide the observed data into K folds, as with usual cross validation, to get an index set $\kappa_{\delta=1}(k)$ for a given k. Similarly, divide the censored data into K folds, obtaining $\kappa_{\delta=0}(k)$. Finally, $\kappa(k)$ is the union of these sets: $\kappa(k) = \kappa_{\delta=1}(k) \cup \kappa_{\delta=0}(k)$. For "real-life data sets like ours", Kohavi (1995) illustrate that 10-fold stratified cross validation performs best.

Repeated cross-validation

The randomness inherent in the cross-validation splits has an effect on the resulting m_{stop}. This is true for boosting in general, but it is true for real-life survival data, especially. In typical survival time data sets one typically has a small effective sample size (number of observed events). We can easily imagine that for two different splits of the data, we can end up with quite different values for m_{stop}. It has been very effectively demonstrated that the split of the folds has a large impact on the choice of m_{stop} (Seibold et al., 2016). Seibold et al. (2016) suggest simply repeating the cross-validation scheme. They show that repeating even 5 times effectively averages out the randomness. In other words, we divide the data into K folds, and repeat this J times. Now let $\kappa(j, k)$ be the k'th fold in the j'th split. We end up with a new estimate for the prediction error,

$$\text{RCV}(m) = \sum_{j=1}^{J} \sum_{k=1}^{K} \sum_{i \in \kappa(j,k)} \text{L}\left((t_i, \delta_i), \theta_m; \mathbf{x}_i\right). \tag{3.9}$$

As before, we choose m_{stop} to be the minimizer of this error,

$$m_{\text{stop}} = \underset{m}{\operatorname{argmin}} \operatorname{RCV}(m). \tag{3.10}$$

In practice, to ensure we find the minimizing m, we let the boosting algorithm run for $m = 1$ to $m = M$, where M is a large number that we are sure will result in a overfitted model.

Chapter 4

First hitting time boost

In this chapter, we propose a component-wise boosting algorithm for fitting the inverse gaussian first hitting time model to survival data.

4.1 Algorithm

We apply the component-wise boosting algorithm 7 with loss function $\rho(\mu, \mathbf{y}0) = -\log \mathrm{L}\, y_0, \mu$. We differentiate the loss function with respect to these two and get .... For more details on the derivation, see A.

Maybe use b instead of y_0, to not get subscript chaos?

We might call this cyclical boosting.

Boost in same

Another way to do this is to only boost one component in each iteration. The component might be corresponding to X, or it might be corresponding to Z.

algo:fhtboost

Algorithm 7 FHT Boost with twodimensional loss function

1. Initialize the n-dimensional vectors $\hat{y}_0^{[0]}, \hat{\mu}^{[0]}$, with offset values, e.g. with $\hat{y}_0^{[0]} = \mathbf{0}, \ldots, \hat{\mu}^{[0]} = \mathbf{0}$. Alternatively, one can use the maximum likelihood estimates as offset values.

2. For both components of the loss function, specify base learners, in particular, a component-wise base learner which can be used for each of the p variables used in $\mathbf{X}$ corresponding to y_0 and the d variables in $\mathbf{Z}$ corresponding to μ. Like earlier, the base learner takes one input variable and has one output variable. Examples include least squares linear regression.

3. Set $m = 0$ and $\nu = 0.1$.

4. Increase m by 1.

   a) If $m > m_{\text{stop},y_0}$, proceed to step 4 e). If not, compute the negative partial derivative $-\frac{\partial \rho}{\partial y_0}$ and evaluate at $\hat{f}^{[m-1]}(X_i, Z_i) = \left(\hat{y}_0^{[m-1]}(X_i), \hat{\mu}^{[m-1]}(Z_i)\right)_{i=1,\ldots,n}$. This yields the negative gradient vector $U_{y_0}^{[m-1]} = \left(U_{i,y_0}^{[m-1]}\right)_{i=1,\ldots,n} := \left(-\frac{\partial}{\partial y_0}\rho\left(Y_i, \hat{f}^{[m-1]}(X_i, Z_i)\right)\right)_{i=1,\ldots,n}$.

   b) Fit the negative gradient vector $U_{y_0}^{[m-1]}$ to each of the p components of $\mathbf{X}$ separately (i.e. to each predictor variable) using the base learners specified in step 2. This yields p vectors of predicted values, where each vector is an estimate of the negative gradient vector $U_{y_0}^{[m-1]}$.

   c) Select the component of $\mathbf{X}$ which best fits $U_{y_0}[m-1]$ according to R^2. Set $\hat{U}_{y_0}^{[m-1]}$ equal to the fitted values of the corresponding best model fitted in the previous step.

   d) Update $\hat{y}_0^{[m-1]} \leftarrow \hat{y}_0^{[m-1]} + \nu \hat{U}_{y_0}^{[m-1]}$.

   e) If $m > m_{\text{stop},\mu}$, proceed to step 4 j). If not, compute the negative partial derivative $-\frac{\partial \rho}{\partial \mu}$ and evaluate at $\hat{f}^{[m-1]}(X_i, Z_i) = \left(\hat{y}_0^{[m-1]}(X_i), \hat{\mu}^{[m-1]}(Z_i)\right)_{i=1,\ldots,n}$. This yields the negative gradient vector $U_{\mu}^{[m-1]} = \left(U_{i,\mu}^{[m-1]}\right)_{i=1,\ldots,n} := \left(-\frac{\partial}{\partial \mu}\rho\left(Y_i, \hat{f}^{[m-1]}(X_i, Z_i)\right)\right)_{i=1,\ldots,n}$.

   f) Fit the negative gradient vector $U_{\mu}^{[m-1]}$ to each of the p components of $\mathbf{Z}$ separately (i.e. to each predictor variable) using the base learners specified in step 2. This yields d vectors of predicted values, where each vector is an estimate of the negative gradient vector $U_{\mu}^{[m-1]}$.

   g) Select the component of $\mathbf{Z}$ which best fits $U_{\mu}[m-1]$ according to R^2. Set $\hat{U}_{\mu}^{[m-1]}$ equal to the fitted values of the corresponding best model fitted in the previous step.

   h) Update $\hat{\mu}^{[m-1]} \leftarrow \hat{\mu}^{[m-1]} + \nu \hat{U}_{\mu}^{[m-1]}$.

   i) Update $\hat{f}^{[m]} \leftarrow \hat{f}^{[m-1]}$.

   j) If $m > \max(m_{\text{stop},y_0}, m_{\text{stop},\mu})$, go to step 5. If not, repeat step 4.

5. Return $\hat{f}^{[m]}$.

Appendices

Appendix A

Appendix 1: Differentiating the IG FHT

appendix

First we have the likelihood,

$$L(y_0,\mu)=\prod_{i=1}^{n}\left(\frac{y_0}{\sqrt{2\pi\sigma^2 t_i^3}}\exp\left[-\frac{(y_0+\mu t_i)^2}{2\sigma^2 t_i}\right]\right)^{\delta_i} \times\left[1-\Phi\left(-\frac{y_0+\mu t_i}{\sqrt{\sigma^2 t_i}}\right)-\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right]^{1-\delta_i}, \tag{A.1}$$

{eq:fht-loglik}

with respect to parameters μ, and y_0. First, note that for any cumulative distribution function F that is symmetric around 0, and for $x\in\mathbb{R}$,

$$F(x)=1-(1-F(x))=1-F(-x), \tag{A.2}$$

and so in particular,

$$\Phi(x)=1-(1-\Phi(x))=1-\Phi(-x), \tag{A.3}$$

and thus we can rewrite (A.1) as

$$L(y_0,\mu)=\prod_{i=1}^{n}\left(\frac{y_0}{\sqrt{2\pi\sigma^2 t_i^3}}\exp\left[-\frac{(y_0+\mu t_i)^2}{2\sigma^2 t_i}\right]\right)^{\delta_i} \times\left[\Phi\left(\frac{y_0+\mu t_i}{\sqrt{\sigma^2 t_i}}\right)-\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right]^{1-\delta_i}. \tag{A.4}$$

{eq:fht-loglik-proper}

It is easier to work with the log likelihood, so we take the log of (A.4) and get

$$l(y_0,\mu)=\sum_{i=1}^{n}\delta_i\left(\ln y_0-\frac{1}{2}\ln\left(2\pi\sigma^2 t_i^3\right)-\frac{(y_0+\mu t_i)^2}{2\sigma^2 t_i}\right) +(1-\delta_i)\ln\left(\Phi\left(\frac{\mu t_i+y_0}{\sqrt{\sigma^2 t_i}}\right)-\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right) \tag{A.5}$$

To make things easier, let us introduce some intermediate functions here. Let

$$\ln f_i(y_0,\mu)=\ln y_0-\frac{1}{2}\ln\left(2\pi\sigma^2 t_i^3\right)-\frac{(y_0+\mu t_i)^2}{2\sigma^2 t_i} \tag{A.6}$$

and

$$S_i(y_0,\mu) = \Phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) - \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right). \tag{A.7}$$

So we get

$$l(y_0,\mu) = \sum_{i=1}^{n} \delta_i \ln f_i(y_0,\mu) + (1-\delta_i)\ln S_i(y_0,\mu) \tag{A.8}$$

Thus we see that the partial derivatives are

$$\frac{\partial}{\partial y_0} l(y_0,\mu) = \sum_{i=1}^{n} \delta_i \frac{\partial}{\partial y_0} \ln f_i(y_0,\mu) + (1-\delta_i)\frac{\frac{\partial}{\partial y_0} S_i(y_0,\mu)}{S_i(\theta)} \tag{A.9}$$

and

$$\frac{\partial}{\partial \mu} l(y_0,\mu) = \sum_{i=1}^{n} \delta_i \frac{\partial}{\partial \mu} \ln f_i(y_0,\mu) + (1-\delta_i)\frac{\frac{\partial}{\partial \mu} S_i(y_0,\mu)}{S_i(\theta)} \tag{A.10}$$

We take these one by one

$$\frac{\partial}{\partial y_0} \ln f_i(y_0,\mu) = \frac{1}{y_0} - \frac{y_0 + \mu t_i}{\sigma^2 t_i} \tag{A.11}$$

$$\frac{\partial}{\partial \mu} \ln f_i(y_0,\mu) = -\frac{y_0 + \mu t_i}{\sigma^2} \tag{A.12}$$

$$\begin{aligned}\frac{\partial}{\partial y_0} S_i(y_0,\mu) = {} & \frac{1}{\sqrt{\sigma^2 t_i}}\phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) + \frac{2\mu}{\sigma^2}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right) \\ & + \frac{1}{\sqrt{\sigma^2 t_i}}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\end{aligned} \tag{A.13}$$

$$\begin{aligned}\frac{\partial}{\partial \mu} S_i(y_0,\mu) = {} & \frac{t_i}{\sqrt{\sigma^2 t_i}}\phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) + \frac{2y_0}{\sigma^2}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right) \\ & - \frac{t_i}{\sqrt{\sigma^2 t_i}}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\end{aligned} \tag{A.14}$$

Hence

$$\begin{aligned}&\frac{\partial}{\partial y_0} l(y_0,\mu) \\ &= \sum_{i=1}^{n}\left(\delta_i\left(\frac{1}{y_0} - \frac{y_0+\mu t_i}{\sigma^2 t_i}\right) + (1-\delta_i)\left[\frac{1}{\sqrt{\sigma^2 t_i}}\phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) + \frac{2\mu}{\sigma^2}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\right.\right. \\ &\left.\left. + \frac{1}{\sqrt{\sigma^2 t_i}}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\right]\left[\Phi\left(\frac{\mu t_i + y_0}{\sqrt{\sigma^2 t_i}}\right) - \exp(-\frac{2y_0\mu}{\sigma^2}\Phi\left(\frac{\mu t_i - y_0}{\sqrt{\sigma^2 t_i}}\right)\right]^{-1}\right)\end{aligned} \tag{A.15}$$

and

$$\frac{\partial}{\partial\mu}l(y_0,\mu)$$
$$=\sum_{i=1}^{n}\left(\delta_i\left(-\frac{y_0+\mu t_i}{\sigma^2}\right)+(1-\delta_i)\left[\frac{t_i}{\sqrt{\sigma^2 t_i}}\phi\left(\frac{\mu t_i+y_0}{\sqrt{\sigma^2 t_i}}\right)+\frac{2y_0}{\sigma^2}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right.\right.$$
$$\left.\left.-\frac{t_i}{\sqrt{\sigma^2 t_i}}\exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right]\left[\Phi\left(\frac{\mu t_i+y_0}{\sqrt{\sigma^2 t_i}}\right)-\exp(-\frac{2y_0\mu}{\sigma^2}\Phi\left(\frac{\mu t_i-y_0}{\sqrt{\sigma^2 t_i}}\right)\right]^{-1}\right) \tag{A.16}$$

Bibliography

aalengjessing2001 Aalen, O. O. and Gjessing, H. K. (2001). Understanding the shape of the hazard rate: a process point of view (with comments and a rejoinder by the authors). *Statist. Sci.*, 16(1):1–22.

buhlmann2006 Bühlmann, P. (2006). Boosting for high-dimensional linear models. *Ann. Statist.*, 34(2):559–583.

buhlmann2007 Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statist. Sci.*, 22(4):477–505.

buhlmann-yu Bühlmann, P. and Yu, B. (2003). Boosting with the l2 loss. *Journal of the American Statistical Association*, 98(462):324–339.

caroni2017 Caroni, C. (2017). *First Hitting Time Regression Models*. John Wiley & Sons, Inc.

chang2010 Chang, Y.-C. I., Huang, Y., and Huang, Y.-P. (2010). Early stopping in l2boosting. *Computational Statistics & Data Analysis*, 54(10):2203 – 2213.

chhikara1988 Chhikara, R. (1988). *The Inverse Gaussian Distribution: Theory: Methodology, and Applications*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis.

adaboost Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

friedman2000 Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Statist.*, 28(2):337–407.

friedman2001 Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232.

hastie2007 Hastie, T. (2007). Comment: Boosting algorithms: Regularization, prediction and model fitting. *Statist. Sci.*, 22(4):513–515.

kearnsvaliant Kearns, M. and Valiant, L. G. (1989). Crytographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 433–444, New York, NY, USA. ACM.

kohavi
Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

lachenbruch
Lachenbruch, P. A. and Mickey, M. R. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10(1):1–11.

leewhitmore2006
Lee, M.-L. T. and Whitmore, G. A. (2006). Threshold regression for survival analysis: Modeling event times by a stochastic process reaching a boundary. *Statist. Sci.*, 21(4):501–513.

lee2010
Lee, M.-L. T. and Whitmore, G. A. (2010). Proportional hazards and threshold regression: their theoretical and practical connections. *Lifetime Data Analysis*, 16(2):196–214.

leewhitmore2004
Lee, M. T., Whitmore, G. A., Laden, F., Hart, J. E., and Garshick, E. (2004). Assessing lung cancer risk in railroad workers using a first hitting time regression model. *Environmetrics*, 15(5):501–512.

mayr14a
Mayr, A., Binder, H., Gefeller, O., and Schmid, M. (2014). The evolution of boosting algorithms. from machine learning to statistical modelling. *Methods of Information in Medicine*, 53(6):419–427.

mayr-hofner
Mayr, A., Hofner, B., and 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(2):178–186.

schmid
Schmid, M., Potapov, S., Pfahlberg, A., and Hothorn, T. (2010). Estimation and regularization techniques for regression models with multidimensional prediction functions. *Statistics and Computing*, 20(2):139–150.

seibold
Seibold, H., Bernau, C., Boulesteix, A.-L., and Bin, R. D. (2016). On the choice and influence of the number of boosting steps.

stogiannis-2013
Stogiannis, D. and Caroni, C. (2013). Issues in fitting inverse gaussian first hitting time regression models for lifetime data. *Communications in Statistics - Simulation and Computation*, 42(9):1948–1960.

tukey
Tukey, J. (1977). *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company.

whitmore1986
Whitmore, G. A. (1986). First-passage-time models for duration data: Regression structures and competing risks. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 35(2):207–219.

threg
Xiao, T., Whitmore, G., He, X., and Lee, M.-L. (2015). The r package threg to implement threshold regression models. *Journal of Statistical Software, Articles*, 66(8):1–16.