Boosting the First-Hitting-Time Regression Model

Vegard Stikbakke

April 11, 2018

Contents

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

2.1 Survival analysis and time-to-event models

sec:survival

In many fields, it is interesting to consider the lifetime of some entity. A lifetime ends when an event occurs. We are usually interested in inferring things about this lifetime, and what it depends upon. In medical fields, this is called survival analysis, while in engineering it is called reliability analysis. In the former case, we consider the lifetime of patients or the length of a hospital stay after some treatment. In the latter, we consider, e.g., the time before a component of a system breaks and must be replaced.

The time-to-event T is a continuous, non-negative random variable $T \sim f(t)$, $t > 0$, for some probability density function f. We are particularly interested in two things related to T:

1. The survival function $S(t)$ – the probability of an individual having survived until time t. Note that $S(t) = 1 - F(t)$, where F is the cumulative density function of f.

2. The hazard function $h(t)$ – the probability of the event happening at time t. Note that this is conditional on surviving until time t, and is defined as $h(t) = \frac{f(t)}{S(t)}$.

Regression

sec:surv-reg

To find out anything interesting, we need to be able to do regression on covariates. Given a sample of n independent observations $\{t_i, \mathbf{x}_i, \delta_i, i = 1, \ldots, n\}$, where individual i has covariates $\mathbf{x}_i$, lifetime t_i and censoring indicator $\delta_i, i = 1, \ldots, n$, which is 1 if the event has happened, and 0 if not. From Caroni 2017, p. 10, the likelihood is given by

$$L(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} f(t_i|\mathbf{x}_i, \boldsymbol{\theta})^{\delta_i} S(t_i|\mathbf{x}_i, \boldsymbol{\theta})^{1-\delta_i} \tag{2.1}$$

{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

$$h(t|x) = h_0(t)g(\mathbf{x}), \tag{2.2}$$

where $h_0(t)$ is a baseline hazard function.

more about baseline hazard?

This property states 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:

$$\frac{h(t_1|x_1)}{h(t_1|x_2)} = \frac{h(t_2|x_1)}{h(t_2|x_2)} \tag{2.3}$$

how to rephrase?

This is a strong assumption to make, and it will rarely be the case in practice (Lee and Whitmore 2010). However, many times Cox regression will work well in practice.

really? or argue why!

2.2 The first hitting time model

sec:fht

Revisiting the examples of the two lifetime settings, it may in both cases be natural to imagine that the event happens as a process reaches a threshold. Then one way to model the time-to-event is to model the process itself, and look at the time it takes for the process to reach this threshold, at which point the event is triggered. Lee and Whitmore 2006 is a thorough review on the first hitting time model, and Caroni 2017 is a book which covers many aspects of it. We continue by describing the first hitting time model.

An first hitting time model has two main components.

1. A stochastic process $\{Y(t), t \in \mathcal{T}, y \in \mathcal{Y}\}$, with $Y(0) = y_0$.
2. A boundary set, $B \subset \mathcal{Y}$, where $y_0 \notin \mathcal{B}$

what did you say about this again?

The first hitting time is the first time the process reaches the boundary set. Formally, the first hitting time is a stochastic variable S, which is defined as

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

Typically, one will consider a process with boundary $B = 0$. The event then occurs if and when the process $\{Y\}$ reaches 0 at $y(S)$. Note that it is possible that $P(S < \infty) < 1$.

justify

The first hitting time model is conceptually appealing and 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 of processes lead to different kinds of distributions for the first hitting time. We now look at a common choice of the process.

Wiener process

sec:wiener

The Wiener process, also known as the standard Brownian motion process, is a process which is continuous in time and space, and has the properties (Caroni 2017, p. 61) that

- $Y(t)$ has independent increments, such that $Y(t_2)-Y(t_1)$ and $Y(t_4)-Y(t_3)$ are independent for any disjoint intervals, and
- for any interval (t_1, t_2),

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

This is a process which will both increase and decrease. However, if we want a monotonic restriction on the movement of the process, we may use a gamma process.

Gamma process

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 inverse gamma. (Lee and Whitmore 2006, p. 503.)

make this into a separate section?

Other choices of processes include Markov chain state models, the Bernoulli process, and the Ornstein-Uhlenbeck process.

2.3 First hitting time regression based on underlying Wiener process

The first hitting time of the Wiener process (section 2.2) follows an inverse Gaussian distribution (derivation in Chhikara 1988, pp. 23-29):

also derive more clearly in appendix?

$$f(t|y_0, \mu, \sigma^2) = \frac{y_0}{\sqrt{2\pi\sigma^2 t^3}} \exp\left[-\frac{(y_0 + \mu t)^2}{2\sigma^2 t}\right] \tag{2.4}$$

{eq:fht-ig}

If μ is positive, $Y(t) \leq 0$ is not certain to occur. Note also that this model is over-parameterized, because Y has an arbitrary scale, so we can without loss of generality set $\sigma^2 = 1$.

more!

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 the variance is $\frac{y_0}{|\mu|^3}$. (Caroni 2017, p. 62.)

The cumulative distribution function of the FHT is (from Xiao et al. 2015, p. 7)

$$F(t|\mu, \sigma^2, y_0) = \Phi\left[(-\frac{y_0 + \mu t)}{\sqrt{\sigma^2 t}}\right] + \exp\left(-\frac{2y_0\mu}{\sigma^2}\right)\Phi\left[\frac{\mu t - y_0}{\sqrt{\sigma^2 t}}\right], \tag{2.5}$$

{eq:cumulative}

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

$$\Phi(x) = \int_{-\infty}^{x} \exp(-y^2/2)/\sqrt{2\pi}\ \mathrm{d}y. \tag{2.6}$$

Regression

We may introduce effects from covariates by allowing μ and y_0 to depend on covariates $\mathbf{x}$. Suitable models are

$$\begin{aligned} \mu &= \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x} \\ \ln y_0 &= \boldsymbol{\gamma}^{\mathrm{T}}\mathbf{z} \end{aligned} \tag{2.7}$$

{eq:coeffs}

where $\boldsymbol{\beta}$ 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 section 2.1, we stated the likelihood of lifetime regression models in (2.1). For an inverse gaussian FHT this then becomes (inserting (2.4) and (2.5) into (2.1), and since $S(t) = 1 - F(t)$)

$$\begin{aligned} L(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = &\left(\frac{y_0}{\sqrt{2\pi\sigma^2 t^3}} \exp\left[-\frac{(y_0 + \mu t)^2}{2\sigma^2 t} \right] \right)^{\delta_i} \\ &\times \left[1 - \Phi\left(-\frac{y_0 + \mu t}{\sqrt{\sigma^2 t}} \right) - \exp\left(-\frac{2y_0\mu}{\sigma^2} \right) \Phi\left(\frac{\mu t - y_0}{\sqrt{\sigma^2 t}} \right) \right]^{1-\delta_i} \end{aligned} \tag{2.8}$$

{eq:fht-lik}

Since we let $\sigma^2 = 1$, this simplifies to

$$\begin{aligned} L(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = &\left(\frac{y_0}{\sqrt{2\pi t^3}} \exp\left[-\frac{(y_0 + \mu t)^2}{2t} \right] \right)^{\delta_i} \\ &\times \left[1 - \Phi\left(-\frac{y_0 + \mu t}{\sqrt{t}} \right) - \exp(-2y_0\mu) \Phi\left(\frac{\mu t - y_0}{\sqrt{t}} \right) \right]^{1-\delta_i} \end{aligned} \tag{2.9}$$

We can now substitute the covariates in (2.7) into this. To find optimal parameters, we use numerical maximum likelihood methods. However, this is only feasible 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.

is this correct?

2.5 The `threg` package

There exists an R package `threg` for fitting regression with inverse gaussian FHT, described in Xiao et al. 2015. We provide a small example here, which is the one described in the help pages of the package.

```
1 library(threg)
2 data("lkr")
3 lkr$f.treatment2=factor(lkr$treatment2)
4 # head(lkr)
5 fit <- threg(Surv(weeks, relapse) ~ f.treatment2|f.treatment2,
      data=lkr)
6 fit
```

Which provides the following output

```
Call:
threg(formula = Surv(weeks, relapse) ~ f.treatment2 | f.treatment2,
    data = lkr)

                          coef  se(coef)         z       p
lny0: (Intercept)    2.0097844 0.1705141 11.786620 0.0e+00
lny0: f.treatment21 -1.2739233 0.2441633 -5.217504 1.8e-07
  mu: (Intercept)   -0.5886165 0.1340126 -4.392246 1.1e-05
  mu: f.treatment21  0.5888365 0.1535081  3.835866 1.3e-04

Log likelihood =-104.64, AIC =217.28
```

Here we fit an inverse gaussian FHT model where

$$\ln y_0 = \mu = \beta_0 + \beta_1 \mathrm{I}(\text{treatment2} = 1)$$

What the `threg` function in the package of the same name does, is essentially to set up the log likelihood and use the numerical optimization function `nlm` to find the optimal parameters.

Recreating

We recreate the above example in plain `R` code.

```
library(threg)
data("lkr")
lkr$f.treatment2=factor(lkr$treatment2)
fit <- threg(Surv(weeks, relapse) ~ f.treatment2|f.treatment2,
    data=lkr)

library(dplyr)
#tbl <- select(.data=lkr, weeks, relapse, f.treatment2)
tbl <- data.frame(lkr$weeks, lkr$relapse, lkr$f.treatment2,
             row_names = c("weeks", "relapse", "f.treatment2"))
names(tbl)[names(tbl) == "lkr.weeks"] <- "weeks"
names(tbl)[names(tbl) == "lkr.relapse"] <- "relapse"
names(tbl)[names(tbl) == "lkr.f.treatment2"] <- "f.treatment2"

to_optimize <- function(params) {
 total_loglikelihood <- 0

 gamma <- params[1:2]
 beta <- params[3:4]

 for (i in 1:n) {
  tbl_i <- tbl[i, ]
  event <- tbl_i$relapse
  t_i <- tbl_i$weeks
  is_treated <- as.integer(tbl_i$f.treatment2)-1
  X_i <- c(1, is_treated)
  y0_i <- exp(sum(gamma*X_i))
  mu_i <- sum(beta*X_i)
```

```
    log_f_i <- log(y0_i) - 0.5*log(2*pi*t_i^3) - ((y0_i +
        mu_i*t_i)^2)/(2*t_i)
    log_S_i <- log(1 - pnorm(-(y0_i+mu_i*t_i)/sqrt(t_i)) -
        exp(-2*y0_i*mu_i)*pnorm((mu_i*t_i-y0_i)/sqrt(t_i)))
    loglik_i <- event*log_f_i + (1 - event)*log_S_i
    total_loglikelihood <- total_loglikelihood + loglik_i
  }
  return(-total_loglikelihood)
}

params_from_threg <- c(2.0098, -1.2739, -0.5886, 0.5886)
threg_value <- -to_optimize(params_from_threg)

initial_params <- c(1, 1, 1, 1)

best <- nlm(to_optimize, initial_params)
params_from_best <- best$estimate
best_value <- -best$minimum
print(best_value)
print(threg_value)
```

We can also inspect the parameters we found and see that we did indeed find the optimal parameters.

CHAPTER 3

Statistical boosting

Boosting

Boosting is one of the most promising methodological approaches for data analysis developed in the last two decades. (Mayr et al. 2014) The history of boosting started with the question posed in 1989 by Kearns and Valiant, working on computational learning theory, of whether any weak learner could be transformed to become also a strong learner. (Kearns and Valiant 1989) A weak classifier is in general defined to be one which is only slightly better than random choice. For regression, it is a bit harder to give a specific definition, but a weak regressor is simple and low dimensional, and does not pick up much of the underlying signal. The answer to the original question is yes, and Schapire and Freund showed this with the AdaBoost algorithm, which constructs a binary classifier. (Freund and Schapire 1996) The algorithm works by iteratively reweighting observations, giving more weight to misclassified observations, and training a new base learner on all observations, using the updated weights. The resulting AdaBoost classifier is a linear combination of these base classifiers. In its original formulation, the classifier does not have interpretable coefficients, and as such it is a so-called black-box algorithm.

3.1 Statistical boosting

sec:sboost

In statistics, however, we are interested in models which are interpretable. We want to estimate the relation between observed predictor variables and the expectation of the response,

$$\mathrm{E}(Y|X = x) = f(x).$$

In addition to using boosting for classification, like in the original AdaBoost, we would also like to use it in more general settings. We therefore extend our discussion to the more general regression scheme, where the outcome variable Y can be continuous. To evaluate a candidate $\hat{f}(x)$, we need to see how well it estimates $f(x)$. This is typically done by choosing a loss function,

$$\mathrm{L}(Y, f(X)), \tag{3.1}$$

{eq:loss}

and calculating the empirical risk, i.e., the average in-sample error over some observed test data set. A typical loss function for regression is the L_2 loss,

$$\mathrm{L}(Y, f(X)) = (Y - f(X))^2$$

The empirical risk is then

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

A possible model for $f(x)$ is the generalized additive model (GAM), in which different effects of single predictors are added,

$$f(x) = \beta_0 + \sum_{i=1}^{p} h_p(x_p). \tag{3.2}$$

{eq:gam}

In 2000, Friedman showed that AdaBoost fits a GAM with a forward stagewise algorithm, for a particular exponential loss function. (J. Friedman, Hastie, and Tibshirani 2000) This provided a way of viewing the successful boosting regime through a statistical lens.

Gradient boosting

Gradient boosting, proposed in 2001, is a boosting scheme which fits a GAM (3.2). (J. H. Friedman 2001) In general, we are interested in finding the function $f(\cdot)$ which minimizes the loss (3.1). From a numerical optimization perspective, this can be seen as

$$\hat{f}(x) = \operatorname*{argmin}_f \mathrm{E}_{Y,X}[\mathrm{L}(Y, f(X))].$$

Gradient boosting does a gradient descent search in function space to find this $\hat{f}(\cdot)$. While the original AdaBoost algorithm iteratively reweights observations, gradient boosting iteratively fits the base-learner to the negative gradient vector $\mathbf{u}^{[m]}$ of the loss function, evaluated at the previous iteration,

$$\mathbf{u}^{[m]} = \left(-\frac{\partial}{\partial f}\mathrm{L}(Y, f)\bigg|_{f=\hat{f}(\cdot)^{[m-1]}}\right).$$

This is the other key point of gradient boosting. Further, often the base learners are one dimensional. Hence,

$$\mathbf{u}^{[m]} = \left(u_1^{[m]}, \cdots, u_p^{[m]}\right),$$

where each component is

$$u_j^{[m]} = -\left(\frac{\partial}{\partial f_j}\mathrm{L}(Y, f)\bigg|_{f=\hat{f}(\cdot)^{[m-1]}}\right)$$

Call k the component which is most negative, i.e.,

$$k = \operatorname{argmin} \mathbf{u}^{[m]},$$

such that $u_k^{[m]}$ is the component in which the gradient of f is steepest. We can then take a gradient descent step in this direction. This gives rise to the component-wise gradient boosting algorithm.

Likelihood-based boosting

To do!

Gradient boosting (re-do)

Gradient boosting was proposed in 2001 (J. H. Friedman 2001), and further refined by Bühlmann and Yu 2003 in 2003. We will here present the gradient boosting framework, first a general algorithm. Assume we have data $\mathbf{X} \in \mathbb{R}^p$ and $Y \in \mathbb{R}$ with some relation $Y \sim f(\mathbf{X})$, $f\colon \mathbb{R}^p \to \mathbb{R}$, which we wish to estimate. We have

$$Y = f(\mathbf{X}) + \epsilon,$$

where ϵ is a random variable with expectation zero. We wish to minimize the expected loss of our distribution,

$$\min \mathrm{E}_{Y,\mathbf{X}}[\mathrm{L}(Y, f(\mathbf{X}))] = \min \mathrm{E}_Y \, \mathrm{E}_{\mathbf{X}}[\mathrm{L}(Y, f(\mathbf{x}))|\mathbf{X} = \mathbf{x}], \tag{3.3}$$

{eq:min-loss}

where L is some meaningful loss function which measures the difference between Y and $f(\mathbf{X})$. In particular, a loss function is 0 if Y is exactly equal $f(\mathbf{X})$, and positive otherwise. To estimate f we typically choose a parameterized model,

$$f(\mathbf{X}) = \boldsymbol{\gamma}^T h(\mathbf{X}), \tag{3.4}$$

where $h(\cdot)$ is some function, and $\boldsymbol{\gamma}$ are parameters to be estimated. For finite observed data points $\{\mathbf{x}_i, Y_i\}_{i=1}^N$ we must estimate the expected loss by the empirical risk,

$$\frac{1}{N} \sum_{i=1}^{N} L(Y_i, f(\mathbf{x}_i)). \tag{3.5}$$

For finite data points and chosen h, there exists β_m which minimizes (3.3),

$$\boldsymbol{\gamma}^* = \min_{\boldsymbol{\gamma}} \frac{1}{N} \sum_{i=1}^{N} L(Y_i, f(\mathbf{x}_i)), \tag{3.6}$$

{eq:min-loss-param}

but estimating this is not necessarily easy. An algorithm which is often used to find an approximate solution $\hat{\boldsymbol{\gamma}}^*$ to 3.6 is gradient descent.

Gradient descent

Gradient descent, or steepest descent, is a greedy iterative numerical optimization algorithm. At each iteration step it improves on the previous solution by going in the direction which improves the loss function the most, which is in the direction of the negative gradient. To avoid overshooting in each step, we perform a line search in the gradient direction, and choose the best step length. The algorithm stops when reaching convergence.

Gradient descent in parameter space

Let $\boldsymbol{\gamma} = \sum_{m=0}^{M} \boldsymbol{\beta}_m$. We start with an initial guess $\boldsymbol{\beta}_0$, e.g. $\boldsymbol{\beta}_0 = 0$. We then carry out steps $m = 1, \ldots, M$, where we find increments $\boldsymbol{\beta}_m$ which improve our existing solution $\boldsymbol{\gamma}_{m-1}$. We continue until some stopping criterion. At each

iteration step in the algorithm, we compute the gradient of the loss with respect to the parameters, evaluated at the current solution,

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \frac{\partial}{\partial \beta_j} \mathrm{E}_{Y,\mathbf{X}}[\mathrm{L}(Y, \boldsymbol{\gamma}_m^T h(\mathbf{X}))] \right\}_{j=1}^{p}. \tag{3.7}$$

We then do a so-called line search to find the optimal step length,

$$\rho_m = \underset{\rho}{\operatorname{argmin}} \, \mathrm{E}_{Y,X}[\mathrm{L}(Y, (\boldsymbol{\gamma}_{m-1} + \rho \mathbf{g}_m)^T h(\mathbf{X}))], \tag{3.8}$$

and choose $\beta_m = \rho_m \mathbf{g}_m$.

Gradient descent in function space

Instead of optimizing a parametric function in parameter space, we can also view the optimization problem (3.3) from a non-parametric perspective. We are then optimizing in functional space, i.e., finding

$$f^* = \min_f \mathrm{E}_{Y,X}[\mathrm{L}(Y, f(\mathbf{X})]. \tag{3.9}$$

{eq:min-loss-func}

In function space there are in theory an infinite number of such f. But in data sets there are only a finite number. We can also here use steepest gradient descent. Following the numerical optimization paradigm as above, we take the solution f^* to (3.9) to be

$$f^* = \sum_{m=0}^{M} h. \tag{3.10}$$

Appendices

Bibliography

buhlmann-yu [1] Bühlmann, P. and Yu, B. "Boosting With the L2 Loss". In: *Journal of the American Statistical Association* 98.462 (2003), pp. 324–339. eprint: https://doi.org/10.1198/016214503000125.

caroni2017 [2] Caroni, C. *First Hitting Time Regression Models*. John Wiley & Sons, Inc., 2017.

chhikara1988 [3] Chhikara, R. *The Inverse Gaussian Distribution: Theory: Methodology, and Applications*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, 1988.

adaboost [4] Freund, Y. and Schapire, R. E. "Experiments with a New Boosting Algorithm". In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML'96. Bari, Italy: Morgan Kaufmann Publishers Inc., 1996, pp. 148–156.

friedman2001 [5] Friedman, J. H. "Greedy function approximation: A gradient boosting machine." In: *Ann. Statist.* 29.5 (Oct. 2001), pp. 1189–1232.

friedman2000 [6] Friedman, J., Hastie, T., and Tibshirani, R. "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)". In: *Ann. Statist.* 28.2 (Apr. 2000), pp. 337–407.

kearnsvaliant [7] Kearns, M. and Valiant, L. G. "Crytographic Limitations on Learning Boolean Formulae and Finite Automata". In: *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*. STOC '89. Seattle, Washington, USA: ACM, 1989, pp. 433–444.

lee2010 [8] Lee, M.-L. T. and Whitmore, G. A. "Proportional hazards and threshold regression: their theoretical and practical connections". In: *Lifetime Data Analysis* 16.2 (Apr. 2010), pp. 196–214.

lee2006 [9] Lee, M.-L. T. and Whitmore, G. A. "Threshold Regression for Survival Analysis: Modeling Event Times by a Stochastic Process Reaching a Boundary". In: *Statist. Sci.* 21.4 (Nov. 2006), pp. 501–513.

mayr14a [10] Mayr, A. et al. "The Evolution of Boosting Algorithms. From Machine Learning to Statistical Modelling". In: *Methods of Information in Medicine* 53.6 (2014), pp. 419–427.

threg [11] Xiao, T. et al. "The R Package threg to Implement Threshold Regression Models". In: *Journal of Statistical Software, Articles* 66.8 (2015), pp. 1–16.