`CoxFlexBoost`: Fitting Structured Survival Models

Benjamin Hofner [1]
Institut für Medizininformatik, Biometrie und Epidemiologie (IMBE)
Friedrich-Alexander-Universität Erlangen-Nürnberg

joint work with Torsten Hothorn and Thomas Kneib
Institut für Statistik
Ludwig-Maximilians-Universität München

useR! 2009 - Rennes

[1] benjamin.hofner@imbe.med.uni-erlangen.de

Data Example - Intensive Care Patients with Severe Sepsis

- **Response:** 90-day survival
- **Predictors:** 14 categorical predictors (sex, fungal infection (y/n), . . .)
  6 continuous predictors (age, Apache II Score, . . .)
- Previous studies showed the presence of linear, non-linear and time-varying effects.

Aims:

- **flexible survival model** for patients suffering from severe sepsis
- identify prognostic factors (at appropriate complexity)

Further Details of the Data-Set:

- **Origin:** Department of Surgery, Campus Großhadern, LMU Munich
- **Period of observation:** March 1993 – February 2005 (12 years)
- **N:** 462 septic patients (180 observations right-censored)

Structured Survival Models

- Cox PH model: $\lambda_i(t) = \lambda(t, \mathbf{x}_i) = \lambda_0(t) \exp(\mathbf{x}_i' \boldsymbol{\beta})$
- **Generalization: Structured Survival Models**

$$\lambda_i(t) = \exp(\eta_i(t))$$

with additive predictor

$$\eta_i(t) = \sum_{l=1}^{L} f_l(\mathbf{x_i}(t)),$$

- Generic representation of covariate effects $f_l(\mathbf{x}_i)$
  a) linear effects: $f_l(\mathbf{x}_i(t)) = f_{l,\text{linear}}(\tilde{x}_i) = \tilde{x}_i \beta$
  b) smooth effects: $f_l(\mathbf{x}_i(t)) = f_{l,\text{smooth}}(\tilde{x}_i)$
  c) time-varying effects: $f_l(\mathbf{x}_i(t)) = f_{l,\text{smooth}}(t) \cdot \tilde{x}_i$ (or $f_l(\mathbf{x}_i(t)) = t\beta \cdot \tilde{x}_i$)

  where $\tilde{x}_i$ is a covariate from $\mathbf{x}_i(t)$.

Note:

c) includes log-baseline ($\tilde{x}_i \equiv 1$)

Estimation

- Flexible terms $f_{l,\text{smooth}}(\cdot)$ can be represented using P-splines (Eilers & Marx, 1996)
- This leads to:

Penalized Likelihood Criterion:

$$\mathcal{L}_{\text{pen}}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\delta_i \eta_i(t_i) - \int_0^{t_i} \exp(\eta_i(t))\, dt \right] - \sum_{l=0}^{L} \text{pen}_l(\boldsymbol{\beta}_l)$$

- NB: this is the **full** log-likelihood

Problem:

Estimation and in particular model choice

- t_i observed survival time
- δ_i indicator for non-censoring
- $\text{pen}_l(\boldsymbol{\beta}_l)$ P-spline penalty for smooth effects

CoxFlexBoost

Aim:

Maximization of the log-likelihood with different modeling alternatives

We use:

- Iterative algorithm called **Likelihood-based Boosting** with component-wise base-learners

Therefore:

- Use one base-learner $g_j(\cdot)$ for each covariate (or each model component) $[\ j \in \{1, \ldots, J\}\ ]$

$\Rightarrow$ Component-wise boosting as is used a means of estimation with intrinsic variable selection and model choice (as we will show now).

CoxFlexBoost

Aim:

Maximization of the log-likelihood with different modeling alternatives

We use:

- Iterative algorithm called **Likelihood-based Boosting** with component-wise base-learners

Therefore:

- Use one base-learner $g_j(\cdot)$ for each covariate (or each model component) $[\ j \in \{1, \ldots, J\}\ ]$

$\Rightarrow$ Component-wise boosting as is used a means of estimation with intrinsic variable selection and model choice (as we will show now).

CoxFlexBoost

Aim:

Maximization of the log-likelihood with different modeling alternatives

We use:

- Iterative algorithm called **Likelihood-based Boosting** with component-wise base-learners

Therefore:

- Use one base-learner $g_j(\cdot)$ for each covariate (or each model component) $[\ j \in \{1, \ldots, J\}\ ]$

$\Rightarrow$ Component-wise boosting as is used a means of estimation with intrinsic variable selection and model choice (as we will show now).

Some Details on CoxFlexBoost

After some initializations, in each boosting iteration m (until $m = m_{\text{stop}}$):

1.) All base-learners $g_j(\cdot)$ (i.e., modeling possibility) are fitted separately (based on penalized MLE).

2.) Choose best fitting base-learner $\hat{g}_{j^*}$ (i.e., the base-learner that maximizes the unpenalized LH)

3.) Add ...

... fraction ν of the fit ($\hat{g}_{j^*}$) to the model

... fraction ν of the parameter estimate ($\boldsymbol{\beta}_{\mathbf{j}^*}$) to the estimation

($\nu = 0.1$ in our case)

What happens then?

(parameters of) previously selected base-learners are treated as a constant in the next iteration

Variable Selection and Model Choice

... is achieved by

- selection of base-learner, i.e., component-wise boosting (steps 1.) & 2.))

and

- early stopping,
  i.e., estimate optimal stopping iteration $\widehat{m}_{\text{stop,opt}}$ via cross validation, bootstrap, ...

- For Variable selection (without model choice):
  Define one base-learner per covariate
  e.g. flexible base-learner with 4 df
- For Variable selection and model choice:
  Define one base-learner per modeling possibility
  But the flexibility must be comparable!
  Otherwise: more flexible base-learners are preferred

Specify Flexibility by Degrees of Freedom

- Specifying the flexibility via df is more intuitive than specifying it via the smoothing parameter κ.
- df can be used to make smooth effects comparable to other modeling components (e.g., linear effects).

Use initial $\widetilde{df}_j$ ($\stackrel{e.g.}{=}$ 4) and solve

$$\mathrm{df}(\kappa_j) - \widetilde{df}_j \stackrel{!}{=} 0$$

for κ_j, where

$$\mathrm{df}(\kappa_j) = \mathrm{trace}\Big[\overbrace{\mathbf{F}_j^{[0]}}^{\text{Fisher matrix}} \big(\underbrace{\mathbf{F}_j^{[0]} + \kappa_j \mathbf{K}_j}_{\text{penalized Fisher matrix}} \big)^{-1} \Big] \qquad \text{(Gray, 1992).}$$

- Problem 1: Not constant over the (boosting) iterations

  But simulation studies showed: No big deviation from the initial $\widetilde{df}_j$

Specify Flexibility by Degrees of Freedom

- Specifying the flexibility via df is more intuitive than specifying it via the smoothing parameter κ.
- df can be used to make smooth effects comparable to other modeling components (e.g., linear effects).

Use initial $\widetilde{df}_j$ ($\overset{e.g.}{=}$ 4) and solve

$$\mathrm{df}(\kappa_j) - \widetilde{df}_j \overset{!}{=} 0$$

for κ_j, where

$$\mathrm{df}(\kappa_j) = \mathrm{trace}\Big[\overbrace{\mathbf{F}_j^{[0]}}^{\text{Fisher matrix}}\big(\underbrace{\mathbf{F}_j^{[0]} + \kappa_j \mathbf{K}_j}_{\text{penalized Fisher matrix}}\big)^{-1}\Big] \qquad \text{(Gray, 1992).}$$

- Problem 1: Not constant over the (boosting) iterations

  But simulation studies showed: No big deviation from the initial $\widetilde{df}_j$

Problem 2

- For P-splines with higher order differences ($d \geq 2$): df > 1 $\quad (\kappa \to \infty)$
- Polynomial of order $d - 1$ remains unpenalized
- **Solution:**

Decomposition for differences of order $d = 2$
(based on Kneib, Hothorn, & Tutz, 2009)

$$f_{\text{smooth}}(x) = \underbrace{\beta_0 + \beta_1 x}_{\text{unpenalized, parametric part}} + \underbrace{f_{\text{smooth,centered}}(x)}_{\text{deviation from polynomial}}$$

- Add unpenalized part as separate, parametric base-learners
- Assign df $= 1$ to the centered effect (and add as P-spline base-learner)
- Analogously for time-varying effects

Technical realization (see Fahrmeir, Kneib, & Lang, 2004):

decomposing the vector of regression coefficients $\boldsymbol{\beta}$ into $(\widetilde{\boldsymbol{\beta}}_{\text{unpen}}, \widetilde{\boldsymbol{\beta}}_{\text{pen}})$ utilizing a spectral decomposition of the penalty matrix

Problem 2

- For P-splines with higher order differences ($d \geq 2$): df > 1 $\quad (\kappa \to \infty)$
- Polynomial of order $d - 1$ remains unpenalized
- **Solution:**

Decomposition for differences of order $d = 2$
(based on Kneib et al., 2009)

$$f_{\text{smooth}}(x) \cdot t = \underbrace{\beta_0 \cdot t + \beta_1 x \cdot t}_{\text{unpenalized, parametric part}} + \underbrace{f_{\text{smooth,centered}}(x) \cdot t}_{\text{deviation from polynomial}}$$

- Add unpenalized part as separate, parametric base-learners
- Assign df $= 1$ to the centered effect (and add as P-spline base-learner)
- Analogously for time-varying effects

Technical realization (see Fahrmeir et al., 2004):

decomposing the vector of regression coefficients $\boldsymbol{\beta}$ into $(\widetilde{\boldsymbol{\beta}}_{\text{unpen}}, \widetilde{\boldsymbol{\beta}}_{\text{pen}})$ utilizing a spectral decomposition of the penalty matrix

Simulation Results (in short)
Properties of CoxFlexBoost

- Good variable selection strategy
- Good model choice strategy if only linear and smooth effects are used
- Selection bias in favor of time-varying base-learners (if present)
  $\Rightarrow$ standardizing time could be a solution
- Estimates are better if decomposition for model choice is used
  (compared to one flexible base-learner with 4 df)

Using CoxFlexBoost - Intro in a Nutshell

A (very) simple example:

- model choice for sampled data with $\lambda = \exp(0.7 \cdot x_1 + x_2^2)$
- `cfboost()` is the main function
- `bols()` represents ordinary least squares base-learners
- `bbs()` represents penalized B-spline base-learners (i.e., P-splines)
- `weights` are used to specify out-of-bag sample (`weights[i] = 0`)

```
R> model <- cfboost(Surv(time, event) ~
                    bols(x1) + bbs(x1, df=1, center=TRUE)
                  + bols(x2) + bbs(x2, df=1, center=TRUE)
                  + bols(x3) + bbs(x3, df=1, center=TRUE),
                  control = boost_control(mstop = 100, risk="oobag"),
                  data = data, weights = weights)
R> model_mstop <- model[mstop(model)]
```

```
R> summary(model_mstop)
   (...)
  Number of selections in 44 iterations:
        bbs(x2):       24
        bols(x1):      18
        bbs(x3):        2
        bbs(x1):        0
        bols(x2):       0
        bols(x3):       0
```

Further base-learners:

- linear time-varying effects $t\,\beta \cdot x_1$:
  `bolsTime(x = time, z = x1)`
- smooth time-varying effects $f_{\text{smooth}}(t) \cdot x_1$ with decomposition:
  `bbsTime(x = time, z = x1, df = 4, center = TRUE)`

Application - Intensive Care Patients with Severe Sepsis (I)

We fitted a component-wise boosting model with P-spline decomposition to achieve model choice and variable selection to the severe sepsis data.

CoxFlexBoost

- selected 10 out of 20 variables + baseline hazard
- used 15 different base-learners (out of 68)

⇒ sparse model

Out of 14 categorical covariates:

- 7 were selected
  - 2 were selected as linear effects
  - 4 were selected as time-varying effects
  - 1 was selected as linear and time-varying effect

Out of 6 continuous covariates:

- 3 were selected
  - 1 with linear effect
  - 2 with linear and time-varying effects

Application - Intensive Care Patients with Severe Sepsis (I)

We fitted a component-wise boosting model with P-spline decomposition to achieve model choice and variable selection to the severe sepsis data.

CoxFlexBoost

- selected 10 out of 20 variables + baseline hazard
- used 15 different base-learners (out of 68)

⇒ sparse model

Out of 14 categorical covariates:

- 7 were selected
  - 2 were selected as linear effects
  - 4 were selected as time-varying effects
  - 1 was selected as linear and time-varying effect

Out of 6 continuous covariates:

- 3 were selected
  - 1 with linear effect
  - 2 with linear and time-varying effects

Application - Intensive Care Patients with Severe Sepsis (I)

We fitted a component-wise boosting model with P-spline decomposition to achieve model choice and variable selection to the severe sepsis data.

CoxFlexBoost

- selected 10 out of 20 variables + baseline hazard
- used 15 different base-learners (out of 68)

⇒ sparse model

Out of 14 categorical covariates:

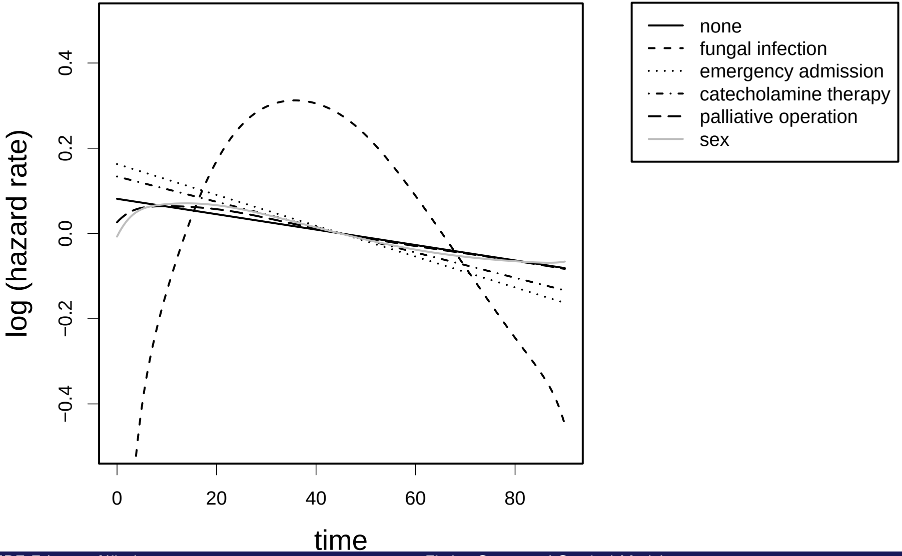
- 7 were selected
  - 2 were selected as linear effects
  - 4 were selected as time-varying effects
  - 1 was selected as linear and time-varying effect

Out of 6 continuous covariates:

- 3 were selected
  - 1 with linear effect
  - 2 with linear and time-varying effects

Application - Intensive Care Patients with Severe Sepsis (II)

Time-varying Effect for Categorical Variables:



Messages "To Go"

R-package `CoxFlexBoost` available on R-forge (Hofner, 2008)

`CoxFlexBoost` ...

- ...allows for variable selection and model choice.
- ...allows for flexible modeling
  - flexible, non-linear effects
  - time-varying effects (i.e., non-proportional hazards)
- ...provides convenient functions to manipulate and show results (`summary()`, `plot()`, `subset()`, ...)
- ...provides built-in function `cv()` to compute $\widehat{m}_{\text{stop,opt}}$ via CV or bootstrap with possible usage of R-package `multicore` (Urbanek, 2009).

References

Hofner, B. (2008). *CoxFlexBoost: Boosting Flexible Cox Models (with Time-Varying Effects).* (R package version 0.6-0)

Hofner, B., Hothorn, T., & Kneib, T. (2008). Variable selection and model choice in structured survival models (Tech. Rep. No. 43). Department of Statistics, Ludwig-Maximilians-Universität München.

Eilers, P. H. C., & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, *11*, 89–121.

Fahrmeir, L., Kneib, T., & Lang, S. (2004). Penalized structured additive regression: A Bayesian perspective. *Statistica Sinica*, *14*, 731–761.

Gray, R. J. (1992). Flexible methods for analyzing survival data using splines, with application to breast cancer prognosis. *Journal of the American Statistical Association*, *87*, 942–951.

Kneib, T., Hothorn, T., & Tutz, G. (2009). Variable selection and model choice in geoadditive regression models. *Biometrics*, *65*, 626–634.

Urbanek, S. (2009). *multicore: Parallel processing of R code on machines with multiple cores or cpus.* (R package version 0.1-3)

Find out more: `http://benjaminhofner.de/`

CoxFlexBoost Algorithm

(i) **Initialization:** Iteration index $m := 0$.

- Function estimates (for all $j \in \{1, \ldots, J\}$):

$$\hat{f}_j^{[0]}(\cdot) \equiv 0$$

- Offset (MLE for constant log hazard):

$$\hat{\eta}^{[0]}(\cdot) \equiv \log\left(\frac{\sum_{i=1}^{n} \delta_i}{\sum_{i=1}^{n} t_i}\right)$$

(ii) **Estimation:** $m := m + 1$.
Fit all (linear/P-spline) base-learners separately

$$\hat{g}_j = g_j(\cdot\,; \hat{\boldsymbol{\beta}}_j),\ \forall j \in \{1, \ldots, J\},$$

by penalized MLE.

Details on pMLE

$$\hat{\boldsymbol{\beta}}_j = \arg\max_{\boldsymbol{\beta}} \mathcal{L}^{[m]}_{j,\text{pen}}(\boldsymbol{\beta})$$

with the penalized log-likelihood (analogously as above)

$$\mathcal{L}^{[m]}_{j,\text{pen}}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\delta_i \cdot (\hat{\eta}_i^{[m-1]} + g_j(x_i(t_i); \boldsymbol{\beta})) - \int_0^{t_i} \exp\left\{ \hat{\eta}_i^{[m-1]}(\tilde{\mathbf{t}}) + g_j(x_i(\tilde{\mathbf{t}}); \boldsymbol{\beta}) \right\} d\tilde{\mathbf{t}} \right] - \text{pen}_j(\boldsymbol{\beta}),$$

with the additive predictor η_i split

- into the estimate from previous iteration $\hat{\eta}_i^{[m-1]}$
- and the current base-learner $g_j(\cdot; \boldsymbol{\beta})$

(iii) **Selection:** Choose base-learner $\hat{g}_{j^*}$ with

$$j^* = \arg \max_{j \in \{1,\dots,J\}} \mathcal{L}_{j,\text{unpen}}^{[m]}(\hat{\beta}_j)$$

(iv) **Update:**

- Function estimates (for all $j \in \{1, \dots, J\}$):

$$\hat{f}_j^{[m]} = \begin{cases} \hat{f}_j^{[m-1]} + \nu \cdot \hat{g}_j & j = j^* \\ \hat{f}_j^{[m-1]} & j \neq j^* \end{cases}$$

- Additive predictor (= fit):

$$\hat{\eta}^{[m]} = \hat{\eta}^{[m-1]} + \nu \cdot \hat{g}_{j^*}$$

with step-length $\nu \in (0, 1]$ (here: $\nu = 0.1$)

(v) **Stopping rule:** Continue iterating steps (ii) to (iv) until $m = m_{\text{stop}}$