Boosting Functional Regression Models

Hands on Tutorial using FDboost – Solutions

Sarah Brockhaus David Rügamer

Department of Statistics, LMU Munich

July 24, 2017

2 Exercises

```
library(FDboost)
?fuelSubset
?emotion
```

2.1 Scalar-on-function regression

Use the `fuelSubset` data to fit different scalar-on-function regression models in the following (i.e., set `data = fuelSubset` in the `FDboost`-call). In each case, use the variable `heatan` as a scalar response. For scalar response regression, the argument `timeformula = NULL`.

1. Fit an additive model with intercept using `1` in the formula argument and a non-linear effect for `h2o` using the `bbs(...)`-base-learner.

```
# make fuelSubset known
data("fuelSubset")

# fit the additive model
mod <- FDboost(heatan ~ 1 + bbs(h2o),
               data = fuelSubset,
               timeformula = NULL,
               control = boost_control(mstop = 100))

# compute empirical risk
er <- cvrisk(mod)
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod, which = 2)
```

2. Extend the model by two linear functional effects for `UVVIS` and `NIR` using the base-learner `bsignal(...)`. First, center the functional covariates to center their effects around zero.

```
fuelSubset$UVVIS <- scale(fuelSubset$UVVIS, scale = FALSE)
fuelSubset$NIR <- scale(fuelSubset$NIR, scale = FALSE)
```

For the argument `s` in `bsignal` use the observed time grid for the respective covariate (`uvvis.lambda`, `nir.lambda`).

```
# fit model
mod <- FDboost(heatan ~ 1 +
                 bbs(h2o, df = 4) +
                 bsignal(UVVIS, s = uvvis.lambda, df = 4) +
                 bsignal(NIR, s = nir.lambda, df = 4),
               data = fuelSubset,
               timeformula = NULL,
               control = boost_control(mstop = 700)
                )

# compute empirical risk
er <- cvrisk(mod)
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod, which = 2:4)
```

3. Fit another model, in which the `bsignal` base-learner is replaced by the functional principal components base-learner `bfpc`.

```
# fit model using bfpc instead of bsignal
mod2 <- FDboost(heatan ~ 1 +
                 bbs(h2o, df = 3) +
                 bfpc(UVVIS, s = uvvis.lambda, df = 3) +
                 bfpc(NIR, s = nir.lambda, df = 3),
               data = fuelSubset,
               timeformula = NULL,
               control = boost_control(mstop = 700)
                )

# compute empirical risk
er2 <- cvrisk(mod2)
# get optimal stopping iteration and plot out-of-bag risk
(ms2 <- mstop(er2))
```

```r
plot(er2)
# set model to optimatl stopping iteration
mod2[ms2]
# look at the summary
summary(mod2)
# plot results
plot(mod2, which = 2:4)
```

4. Use `predict` to compute predictions for the previous two models and compare the predictions.

```r
# compute predictions
pred1 <- predict(mod)
pred2 <- predict(mod2)

# compare predictions
plot(pred1 ~ pred2)
abline(0, 1)
```

5. Advanced Exercise: Compute bootstrap intervals for the estimated coefficients using the `bootstrapCI` function and plot the intervals using the `plot` method.

```r
# compute bootstrap intervals
bci <- bootstrapCI(mod, B_inner = 10, type_inner = "kfold")
# plot intervals
plot(bci)
```

2.2 Function-on-function regression

Use the `emotion` data to fit different function-on-function regression models in the following (i.e., set `data = emotion` in the `FDboost`-call). In each case, use the variable `EMG` as a functional response. For functional response regression, use the argument `timeformula = ~ bbs(t, df = 3)`.

1. Fit a pure intercept model.

```r
# make fuelSubset known
data("emotion")

# fit the pure intercept model
mod <- FDboost(EMG ~ 1,
               data = emotion,
               timeformula = ~ bbs(t, df = 3),
               control = boost_control(mstop = 100))

# compute empirical risk
er <- applyFolds(mod)
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
```

```
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod)
```

2. Fit a function-on-scalar model with a subject effect and an effect for `power` using the base-learner `bolsc(...)` in both cases.

```
# fit the function-on-scalar model
mod <- FDboost(EMG ~ 1 +
                 bolsc(subject, df = 2) +
                 bolsc(power, df = 2),
               data = emotion,
               timeformula = ~ bbs(t, df = 3),
               control = boost_control(mstop = 100))

# compute empirical risk
er <- applyFolds(mod)
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod)
```

3. In order to reduce computational burden for the following model fits, create a subset of the `emotion` data as follows and make yourself familiar with the data structure.

```
# define subset for a certain game condition
subset <- emotion$control == "high" &
  emotion$game_outcome == "gain" &
  emotion$power == "low"
emotionHGL <- list()

# subset scalar variables
emotionHGL$subject <- emotion$subject[subset]

# subset functional variables
emotionHGL$EMG <- emotion$EMG[subset,]
emotionHGL$EEG <- emotion$EEG[subset,]
# center the functional covariate per time-point
```

```r
emotionHGL$EEG <- scale(emotionHGL$EEG, scale = FALSE)

# keep the evaluation points
emotionHGL$s <- emotionHGL$t <- emotion$t
```

4. Use the `emotionHGL` subset to fit a function-on-function regression model with a signal effect for `EEG` using the base-learner `bsignal(EEG, s = s)`.

```r
# fit the function-on-function model
# with signal effect for EEG
mod <- FDboost(EMG ~ 1 +
                 bsignal(EEG, s = s),
               data = emotionHGL,
               timeformula = ~ bbs(t, df = 3),
               control = boost_control(mstop = 100))

# compute empirical risk
er <- applyFolds(mod, folds =
                   cv(rep(1, length(unique(mod$id))), type =
                        "kfold"))
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod)
```

5. Replace the signal effect with a concurrent effect using the base-learner `bconcurrent(EEG, s = s, time = t)`.

```r
# fit the function-on-function model
# with concurrent effect for EEG
mod <- FDboost(EMG ~ 1 +
                 bconcurrent(EEG, s = s, time = t),
               data = emotionHGL,
               timeformula = ~ bbs(t, df = 3),
               control = boost_control(mstop = 100))

# compute empirical risk
er <- applyFolds(mod, folds =
                   cv(rep(1, length(unique(mod$id))), type =
                        "kfold"))
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
```

```r
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod)
```

6. Replace the concurrent effect with a historical effect using the base-learner `bhist(EEG, s = s, time = t, limits = function(s,t) s <= t)`.

```r
# fit the function-on-function model
# with historical effect for EEG
mod <- FDboost(EMG ~ 1 +
                 bhist(EEG, s = s, time = t,
                       limits = function(s,t) s <= t),
               data = emotionHGL,
               timeformula = ~ bbs(t, df = 3),
               control = boost_control(mstop = 100))

# compute empirical risk
er <- applyFolds(mod, folds =
                   cv(rep(1, length(unique(mod$id))), type =
                        "kfold"))
# get optimal stopping iteration and plot out-of-bag risk
(ms <- mstop(er))
plot(er)
# set model to optimatl stopping iteration
mod[ms]
# look at the summary
summary(mod)
# plot results
plot(mod)
```