Viscosity: function-on-scalar regression

Sarah Brockhaus

1 Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstraße 33, D-80539 München, Germany
sarah.brockhaus@stat.uni-muenchen.de

1 Descriptive analysis

Load FDboost package and useful functions for plotting.

Load data and choose the time-interval.

```r
> # load("viscosity.RData")
> data(viscosity)
> str(viscosity)

List of 7
$ visAll : AsIs [1:64, 1:132] 41.5 25.2 63.7 35.6 17.8 12.3 38.6 22 18.2 36 ...
$ timeAll: num [1:132] 11 13 15 17 19 21 23 25 27 29 ...
$ T_C   : Factor w/ 2 levels "low","high": 1 1 2 2 2 2 1 1 1 1 ...
$ T_A   : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 1 1 ...
$ T_B   : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 1 2 ...
$ rspeed : Factor w/ 2 levels "low","high": 1 2 1 2 1 2 2 1 2 1 ...
$ mflow : Factor w/ 2 levels "low","high": 2 1 1 2 1 2 1 2 2 1 ...
```

> ## set time-interval that should be modeled
> interval <- "509"
> ## model time until "interval"
> end <- which(viscosity$timeAll==as.numeric(interval))
> viscosity$vis <- log(viscosity$visAll[,1:end])
> viscosity$time <- viscosity$timeAll[1:end]
> ## set up interactions by hand
> vars <- c("T_C", "T_A", "T_B", "rspeed", "mflow")
> for(v in 1:length(vars)){
+ for(w in v:length(vars))
+ viscosity[[paste(vars[v], vars[w], sep="_")]] <- factor(
+   (viscosity[[vars[v]]]:viscosity[[vars[w]]]=="high:high")*1)
+ }
> #str(viscosity)
> names(viscosity)
Figure 1: Viscosity over time with temperature of tools ($T_C$) and temperature of resin ($T_A$) color coded.

2 Model with all main effects and interactions of first order

Fit model with all main effects and interactions.
> set.seed(1911)
> modAll <- FDboost(vis ~ 1
+    + bols(T_C) # main effects
+    + bols(T_A)
+    + bols(T_B)
+    + bols(rspeed)
+    + bols(mflow)
+    + bols(T_C_T_A) # interactions T_WZ
+    + bols(T_C_T_B)
+    + bols(T_C_rspeed)
+    + bols(T_C_mflow)
+    + bols(T_A_T_B) # interactions T_A
+    + bols(T_A_rspeed)
+    + bols(T_A_mflow)
+    + bols(T_B_rspeed) # interactions T_B
+    + bols(T_B_mflow)
+    + bols(rspeed_mflow), # interactions rspeed
+    + timeformula="bbs(time, lambda=100),
+    + numInt="Riemann", family=QuantReg(),
+    + offset=NULL, offset_control = o_control(k_min = 10),
+    + data=viscosity, check0=FALSE,
+    + control=boost_control(mstop = 100, nu = 0.2))

Get optimal stopping iteration using bootstrap over curves.

> set.seed(1911)
> folds <- cv(weights=rep(1, modAll$ydim[1]), type="bootstrap", B=10)
> cvmAll <- suppressWarnings(validateFDboost(modAll, folds = folds,
+    + getCoefCV=FALSE,
+    + grid=seq(10, 500, by=10), mc.cores=10))
> mstop(cvmAll) # 180
> # modAll <- modAll[mstop(cvmAll)]
> # summary(modAll)
> # cvmAll

Do model selection using stability selection.

> set.seed(1911)
> folds <- cvMa(ydim=modAll$ydim, weights=model.weights(modAll),
+    + type = "subsampling", B = 50)
> stabsel_parameters(q=5, PFER=2, p=16, sampling.type = "SS")
> sel1 <- stabsel(modAll, q=5, PFER=2, folds=folds, grid=1:100,
+    + sampling.type="SS", mc.cores=10)
> sel1
> # selects effects T_C, T_A, T_C_T_A

The effects $T_A$, $T_B$ and their interaction are selected into the model.
3 Model with selected effects

Estimate the model containing only the selected effects $T_C$, $T_A$, and their interaction.

```r
> set.seed(1911)
> mod1 <- FDboost(vis ~ 1 + bols(T_C) + bols(T_A) + bols(T_C_T_A),
+  timeformula="bs(time, lambda=100),
+  numInt="Riemann", family=QuantReg(), check0=FALSE,
+  offset=NULL, offset_control = o_control(k_min = 10),
+  data=viscosity, control=boost_control(mstop = 200, nu = 0.2))
> mod1 <- mod1[430]
```

Find the optimal stopping iteration.

```r
> set.seed(1911)
> folds <- cv(weights=rep(1, mod1$ydim[1]), type="bootstrap", B=10)
> cvm1 <- suppressWarnings(validateFDboost(mod1, folds = folds,
+  getCoefCV=FALSE,
+  grid=seq(10, 500, by=10), mc.cores=10))
> mstop(cvm1) # 430
> mod1 <- mod1[mstop(cvm1)]
> # summary(mod1)
```

Center all coefficient functions at each timepoint, yielding the following model:

$$\text{median}\{\log(\text{vis}_i(t)) | x_i\} = \beta_0(t) + T_{Ai} \beta_A(t) + T_{Ci} \beta_C(t) + T_{ACi} \beta_{AC}(t),$$

where $\text{vis}_i(t)$ is the viscosity of observation $i$ at time $t$, $T_{Ai}$ and $T_{Ci}$ are the temperatures of resin and of tools, respectively, each coded as -1 for the lower and 1 for the higher temperature. The interaction $T_{ACi}$ is 1 if both temperatures are in the higher category and -1 otherwise.

```r
> # set up dataframe containing systematically all variable combinations
> newdata <- list(T_C=factor(c(1,1,2,2), levels=1:2, labels=c("low","high")),
+  T_A=factor(c(1, 2, 1, 2), levels=1:2, labels=c("low","high")),
+  T_C_T_A=factor(c(1, 1, 2, 1), levels=c(1, 1, 1, 1), time=mod1$yind)
> intercept <- 0
> ## effect of T_C
> pred2 <- predict(mod1, which=2, newdata=newdata)
> intercept <- intercept + colMeans(pred2)
> pred2 <- t(t(pred2)-intercept)
> ## effect of T_A
> pred3 <- predict(mod1, which=3, newdata=newdata)
> intercept <- intercept + colMeans(pred3)
> pred3 <- t(t(pred3)-colMeans(pred3))
```
Plot the centered coefficient functions.

> pdf("visMod.pdf")
> par(mfrow=c(1,1), mar=c(3, 3, 1, 2), cex=1.5)
> mycol <- gray(seq(0, 0.5, l=3), alpha=0.8)
> funplotLogscale(mod1$yind, pred2[3:4,], col=mycol[1], ylim=c(-0.5,6), lty=2, lwd=2)
> lines(mod1$yind, pred3[2,], col=mycol[2], lty=3, lwd=2)
> lines(mod1$yind, pred4[4,], col=mycol[3], lty=4, lwd=2)
> legend("topright", lty=2:4, lwd=2, col=mycol,
+     legend=c("effect T_C high","effect T_A high","effect T_C, T_A high"))
> dev.off()

null device

Figure 2: Viscosity over time and estimated coefficient functions. On the left hand side the viscosity measures are plotted over time with temperature of tools ($T_C$) and temperature of resin ($T_A$) color-coded. On the right hand side the coefficient functions are plotted.