

Regression III
Lecture 8: Resampling
Bootstrapping, Jackknifing, Cross-Validation
(Addendum)

Dave Armstrong

University of Wisconsin – Milwaukee
Department of Political Science

e: armstrod@uwm.edu

w: www.quantoid.net/ICPSR.php

Bootstrap Test of Non-linearity

We suggested before that the F-test when dealing with smoothers is only approximate for a couple of reasons:

- Distributional assumptions about the difference between $\hat{f}(x)$ and $f(x)$ may be unwarranted or unjustified.
- The degrees of freedom are only an approximation based on (various) analogies to the linear model.
- Yet to be discussed - when GCV and back-fitting are used to estimate the smoothing parameter on the penalized fit function, little is known about the distribution of the test statistic under H_0 .

Bootstrapping can be a way of providing approximately the “right” answer in the face of these problems.

Steps to Tests for Linearity

(from Keele's "Semiparametric Regression for the Social Sciences" p. 191).

1. First, estimate both the linear model and GAM such that:

$$\hat{y}_i = \gamma x_i + \beta Z$$

$$\tilde{y}_i = f(x_i) + \beta Z$$

2. Calculate t , the test statistic (here, the difference in model deviances).
3. Calculate the parametric model residuals $\hat{e}_i = y_i - \hat{y}_i$. And take bootstrap samples $e_1^*, e_2^*, \dots, e_n^*$ from the original vector of residuals.
4. Calculate the bootstrap response:

$$y_i^* = \hat{y}_i + \hat{e}_i^*$$

and use the original predictors x_i and Z to estimate both the parametric and non-parametric models of y_i^* on x_i and Z . Use these results to re-calculate t^* and do this R times.

5. Calculate the new p-value:

$$\frac{[1 + \#(t^* \geq t)]}{(B + 1)}$$

Test in R

```
> library(mgcv)
> library(boot)
> library(car)
> mod1 <- gam(prestige ~ log(income) + poly(women, 2) +
+ poly(education, 2), data=Prestige)
> mod2 <- gam(prestige ~ s(income, bs="cr") + poly(women, 2) +
+ poly(education, 2), data=Prestige)
> orig.t <- deviance(mod1) - deviance(mod2)
> resids <- mod1$residuals
> yhat <- mod1$fitted
> test.fun <- function(dat, inds){
+ assign(".inds", inds, envir=.GlobalEnv)
+ boot.e <- resids[.inds]
+ boot.y <- yhat + boot.e
+ tmp.mod1 <- gam(boot.y ~ log(income) +
+ poly(women, 2) + poly(education, 2),
+ data=dat)
+ tmp.mod2 <- gam(boot.y ~ s(income, bs="cr") +
+ poly(women, 2) + poly(education, 2),
+ data=dat)
+ remove(".inds", envir=.GlobalEnv)
+ deviance(tmp.mod1) - deviance(tmp.mod2)
+ }
```

Looking at the Results

```
> boot.t <- boot(Prestige, test.fun, R=1000)
> (sum(boot.t$t >= orig.t)+1)/(1000+1)
```

```
[1] 0.2767233
```

```
> boot.ci(boot.t, type=c("perc", "bca"))
```

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

Based on 1000 bootstrap replicates

CALL :

```
boot.ci(boot.out = boot.t, type = c("perc", "bca"))
```

Intervals :

Level	Percentile	BCa
95%	(-118.0, 700.1)	(16.1, 1136.7)

Calculations and Intervals on Original Scale

Warning : BCa Intervals used Extreme Quantiles

Some BCa intervals may be unstable

Cross-validating Span in Loess

We could use cross-validation to tell us something about the span in our Loess model.

1. First, split the sample into K groups (usually 10).
2. For each of the $k = 10$ groups, estimate the model on the other 9 and get predictions for the omitted groups observations. Do this for each of the 10 subsets in turn.
3. Calculate the CV error: $\frac{1}{n} \sum (y_i - \hat{y}_i)^2$
4. Potentially, do this lots of times and average across the CV error.

```
> library(foreign)
> dat <- read.dta("~/Desktop/ICPSR_slides/Lecture 6/jacob.dta")
> dat$perot <- (dat$perotvote - min(dat$perotvote))/
+ diff(range(dat$perotvote))
> dat$chal <- (dat$chal_vote - min(dat$chal_vote))/
+ diff(range(dat$chal_vote))
> lo.mod <- loess(chal ~ perot, data=dat, span=.75)
```

CV Loess in R

I took the code from the `cv.glm` function from the `boot` package and repurposed it for loess. This is in `cv.lo.R` which is on the course webpage.

```
> source("cv.lo.R")
> out.cv <- matrix(ncol=2, nrow=9)
> out.cv[1,] <- cv.lo(dat, "chal", update(lo.mod, span=.1),
+ numiter=25, K=10)
> out.cv[2,] <- cv.lo(dat, "chal", update(lo.mod, span=.2),
+ numiter=25, K=10)
> out.cv[3,] <- cv.lo(dat, "chal", update(lo.mod, span=.3),
+ numiter=25, K=10)
> out.cv[4,] <- cv.lo(dat, "chal", update(lo.mod, span=.4),
+ numiter=25, K=10)
> out.cv[5,] <- cv.lo(dat, "chal", update(lo.mod, span=.5),
+ numiter=25, K=10)
> out.cv[6,] <- cv.lo(dat, "chal", update(lo.mod, span=.6),
+ numiter=25, K=10)
> out.cv[7,] <- cv.lo(dat, "chal", update(lo.mod, span=.7),
+ numiter=25, K=10)
> out.cv[8,] <- cv.lo(dat, "chal", update(lo.mod, span=.8),
+ numiter=25, K=10)
> out.cv[9,] <- cv.lo(dat, "chal", update(lo.mod, span=.9),
+ numiter=25, K=10)
```

Results

```
> s <- seq(.1,.9, by=.1)
> pdf("cvres.pdf", height=4, width=6)
> plot(s, out.cv[,1], xlab = "Span", ylab="CV Error",
+      ylim=range(c(out.cv)), type="b", pch=16)
> lines(s, out.cv[,2], type="b", col="red", pch=15)
> legend("topright", c("Raw", "Corrected"),
+      pch=c(16,15), col=c("black", "red"),
+      inset=.01)
> invisible(dev.off())
```

