LSIRM Statistical/Machine Learning
Cross-validation

Dave Armstrong

University of Western Ontario
Department of Political Science

e: dave.armstrong@uwo.ca
w: www.quantoid.net/teachwlu/

Cross-Validation (1)

- If no two observations have the same Y, a p-variable model fit to $p + 1$ observations will fit the data precisely
- this implies, then, that discarding much of a dataset can lead to better fitting models
  - Of course, this will lead to biased estimators that are likely to give quite different predictions on another dataset (generated with the same DGP)
- Model validation allows us to assess whether the model is likely to predict accurately on future observations or observations not used to develop this model
- Three major factors that may lead to model failure are:
  1. Over-fitting
  2. Changes in measurement
  3. Changes in sampling
- External validation involves retesting the model on new data collected at a different point in time or from a different population
- Internal validation (or cross-validation) involves fitting and evaluating the model carefully using only one sample

Cross-Validation (2)

- A basic, but powerful, tool for statistical model building
- Cross-validation is similar to bootstrapping in that it resamples from the original data
- The basic form involves randomly dividing the sample into two subsets:
  - The first subset of the data (screening sample) is used to select or estimate a statistical model
  - The second subset is then used to test the findings
- Can be helpful in avoiding capitalizing on chance and over-fitting the data - i.e., findings from the first subset may not always be confirmed by the second subsets
- Cross-validation is often extended to use several subsets (either a preset number chosen by the researcher or leave-one-out cross-validation)

Cross-Validation (3)

- The data are split into k subsets (usually $3 \leq k \leq 10$), but can also use leave-one-out cross-validation
- Each of the subsets are left out in turn, with the regression run on the remaining data
- Prediction error is then calculated as the sum of the squared errors:

$$RSS = \sum (Y_i - \hat{Y}_i)^2$$

- We choose the model with the smallest average "error"

$$MSE = \frac{\sum (Y_i - \hat{Y}_i)^2}{n}$$

- We could also look to the model with the largest average R^2

Cross-Validation (4)

How many observations should I leave out from each fit?

- There is no rule on how many cases to leave out, but Efron (1983) suggests that grouped cross-validation (with approximately 10% of the data left out each time) is better than leave-one-out cross-validation

Number of repetitions

- Harrell (2001:93) suggests that one may need to leave $\frac{1}{10}$ of the sample out 200 times to get accurate estimates

Cross-validation does not validate the complete sample

- External validation, on the other hand, validates the model on a new sample
- Of course, limitations in resources usually prohibits external validation in a single study

Cross-Validation in **R**

- Cross-validation is done easily using the `cv.glm` function in the `boot` package

```
library(boot)
dat <- read.csv("http://www.quantoid.net/files/reg3/weakliem.txt",
  header=T)
dat <- dat[-c(25,49), ]
mod1 <- glm(secpay ~ poly(gini, 3)*democrat, data=dat)
mod2 <- glm(secpay ~ gini*democrat, data=dat)

deltas <- NULL
for(i in 1:25){
deltas <- rbind(deltas, c(
  cv.glm(dat, mod1, K=5)$delta,
  cv.glm(dat, mod2, K=5)$delta)
)}
colMeans(deltas)

## [1] 0.008278339 0.007742435 0.005861031 0.005715323
```


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.

```
set.seed(1)
n <- 400
x <- 0:(n-1)/(n-1)
f <- 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
y <- f + rnorm(n, 0, sd = 2)
tmp <- data.frame(y=y, x=x)
lo.mod <- loess(y ~ x, data=tmp, span=.75)
```

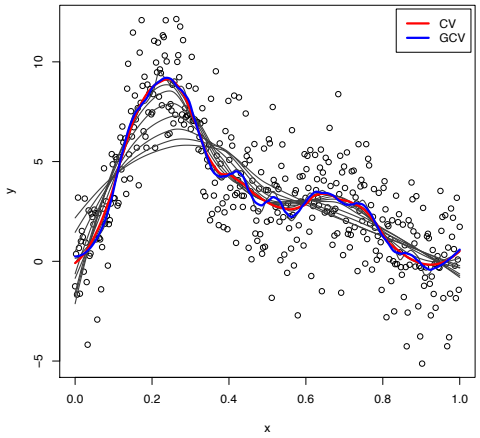

Minimizing CV Criterion Directly

```
library(DAMisc)
best.span <- optimize(cv.lo2, c(.05,.95), form=y ~ x, data=tmp,
            numiter=5, K=10)
best.span

## $minimum
## [1] 0.2271807
##
## $objective
## [1] 3.912508
```

There is also a canned function in `fANCOVA` that optimizes the span via AICc or GCV.

```
library(fANCOVA)
best.span2 <- loess.as(tmp$x, tmp$y, criterion="gcv")
best.span2$pars$span

## [1] 0.1483344
```

The Curve



Manually Cross-Validating λ in the Y-J Transform

Sometimes, optimizing the cross-validation criterion fails.

- Randomness in the CV procedure can produce a function that has several local minima.
- You could force the same random split at every evaluation by hand-coding the CV, but this might not be the best idea.
- If optimization of the CV criterion fails, you could always do it manually.

Example

```
cvoptim_yj <- function(pars, form, data, trans.vars, K=5, numiter=10){
    require(boot)
        require(VGAM)
    form <- as.character(form)
    for(i in 1:length(trans.vars)){
        form <- gsub(trans.vars[i], paste("yeo.johnson(", trans.vars[i],
                ",", pars[i], ")", sep=""), form)
    }
    form <- as.formula(paste0(form[2], form[1], form[3]))
    m <- glm(as.formula(form), data, family=gaussian)
    d <- lapply(1:numiter, function(x)cv.glm(data, m, K=K))
    mean(sapply(d, function(x)x$delta[1]))
}

lams <- seq(0,2, by=.1)
s <- sapply(lams, function(x)cvoptim_yj(x, form=prestige ~ income + education + women,
        data=Prestige, trans.vars="income", K=3))
plot(s ~ lams, type="l", xlab=expression(lambda), ylab="CV Error")
```


Figure