Chapter 9 coding

DJM

8 March 2018

Mid-semester evaluation

hrs	Freq
0-3	0
3-6	7
6-10	3
10-15	3
15+	0

Comments:

- More math, less coding (3)
- More coding, less math (6)
- Text helps (3)
- Text doesn't help (1)
- Prefer group HW (5)
- Detest group HW (3)

Things to do:

- More worked examples (like hedge fund, California)
- Enforce HW participation, too much HW free-loading.
- Make you read things **before** class. (But people don't care much for RRs.)
- A request for in-class coding.

Exam comments

Excellent - 4
Very good - 4
Good - 4
Ok, but probably see me - 2
I need to see you yesterday - 2

- What to look for in diagnostic plots:

1. QQ plot - Normality vs skewness vs heavy tails.
2. Residuals vs predictors - dependence between ϵ and x (patterns or heteroskedasticity).
3. Residuals vs Fitted - patterns are bad

- Residuals always have mean zero (unless you didn't estimate an intercept)
- Normality is not the same as mean zero
- Bootstrap: if heteroskedasticity or patterns or other evidence that the model is wrong, use non-parametric. If model is true and no heteroskedasticity, parametric is ok.
- `npreg` estimates a function $\widehat{f}(x_1, ..., x_p)$. It's arbitrary. It will find interactions or handle factors automatically, but there aren't any coefficients!
- Recall the differentiation we did on the board: this is why that plot at the end is useful.
- To include `x1^2` in `lm` (or another model) use `I(x1^2)`.

GAMs

- Here we introduce the concept of GAMs (**G** eneralized **A** dditive **M** odels)
- The basic idea is to imagine that the response is the sum of some functions of the predictors:

$$\mathbb{E}\left[Y_i \mid X_i = x_i\right] = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip}).$$

- Note that OLS **is** a GAM (take $f_j(x_{ij}) = \beta_j x_{ij}$):

$$\mathbb{E}\left[Y_i \mid X_i = x_i\right] = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}.$$

- The algorithm for fitting these things is called "backfitting":
  1. Center Y and X.
  2. Hold f_k for all $k \neq j$ fixed, and regress f_j on the partial residuals using your favorite smoother.
  3. Repeat for $1 \leq j \leq p$.
  4. Repeat steps 2 and 3 until the estimated functions "stop moving" (iterate)
  5. Return the results.

Results

- We will code it today.
- There are two R packages that do this for us. I find mgcv easier.

The code

```r
backfitlm <- function(y, X, max.iter=100, small.enough=1e-4, track=FALSE){
  # This function performs linear regression by "backfitting"
  # Inputs: y - the response
  #         X - the design matrix
  #         max.iter - the maximum number of loops through the predictors (default to 100)
  #         small.enough - if the mse changes by less than this, terminate (default to 1e-6)
  X = as.matrix(X)
  p = ncol(X) # how many covariates?
  n = nrow(X) # how many observations
  betas = double(p) # create a vector for our estimated coefficients (all zeros now)
  preds = matrix(0, n, p) # a matrix to hold the partial predictions of each covariate
  pres = matrix(y, n, p) # partial residuals begin as y (since we are predicting with 0)
  iter = 0 # initialize our iteration
  mse = mean(y^2) # initialize the MSE to SSTo
  conv = FALSE # initialize the convergence check to FALSE
  while(!conv && (iter < max.iter)){ # enter the loop, check conditions (note && not &)
    iter = iter + 1 # update the iteration count
    for(j in 1:p){ # loop over all predictors
      pres[,j] = y - rowSums(preds[,-j]) # partial residuals (ignoring current predictor)
      ## same as X[,-j] %*% betas[-j], same as apply(preds[,-j],1,sum)
      mod = lm(pres[,j] ~ X[,j]-1) # regress current predictor on partial residuals, no intercept, if g
      ## mod = gam(pres[,j]~X[,j]) # if we want a gam instead
      betas[j] = coefficients(mod) # get out the single coefficient, if gam, remove this line
      preds[,j] = fitted(mod) # update the predictions from this column
    }
```

```r
    msenew = sqrt(mean((y - rowSums(preds)))^2) # get the updated MSE after a pass
    conv = (abs(mse-msenew)<small.enough) # check how different our MSE was from previous
    mse = msenew # save the new MSE
    if(track) cat(iter/max.iter, " mse = ", mse, "\n")
  }
  return(list(bhat=betas, pres=pres, preds = preds, mse=mse)) # return our coefficients
}
```

Testing…

- Generate a design matrix X with 100 observations and $p = 10$ covariates and a response variable y using a linear model.
- Test the (now complete) `backfitlm` on this data and compare the results to `lm`. Should there be an intercept in either version?

```r
set.seed(03-08-2017)
n = 100
p = 10
X = matrix(runif(n*p,-1,1), n)
b = 10:1 # true betas
y = X %*% b + rnorm(n, sd=.5)
bhat.lm = coef(lm(y~X-1)) # no intercept
bhat.bf = backfitlm(y,X)$bhat # also no intercept
round(rbind(bhat.lm,bhat.bf),3)
```

```
##            X1    X2    X3   X4    X5    X6    X7    X8    X9   X10
## bhat.lm 9.893 8.985 8.008 6.95 6.004 5.002 4.029 2.905 2.005 0.866
## bhat.bf 9.894 8.986 8.007 6.95 6.004 5.002 4.029 2.905 2.005 0.866
```

Notice that the estimated coefficients are exactly the same. I didn't generate data with an intercept, so I didn't let `lm` estimate one. You could have though. You just need to include a column of ones in the `X` matrix.