Backfitting

DJM

11 March 2017

Backfitting

```
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 = npreg(pres[,j]~X[,j], tol=1e-4,ftol=1e-4) # 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
    }
    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
}
```

- 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?

```
set.seed(03-09-2017)
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 10.096 8.974 7.914 6.927 5.952 4.872 4.014 2.984 1.765 0.995
## bhat.bf 10.096 8.974 7.914 6.927 5.952 4.872 4.014 2.984 1.765 0.995
```

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.