

STAT361 Laboratory for Advanced R for Data Science

Lab 7

Sidi Wu

March 9/10, 2023

Department of Statistics and Actuarial Science
Simon Fraser University

Update & Test 'fwd_stepwise()'
(MARS - Algorithm 2)

Objective

- Modify the `recpart_fwd()` function into `fwd_stepwise()` function, based on Algorithm 2.
- Test `fwd_stepwise()` against the given test data to verify whether the implementation is correct or not.

Step 1

- Pull Stat360 class repository and look for 'Testfiles' within the 'Lab7' directory. You should see,
 - mctest.RData
 - xtest.RData
 - fwdtest.RData
- Copy the R data files into your working directory and load R data file as follows,
 - `load("../Testfiles/mctest.RData")`
 - `load("../xtest.RData")`
 - `load("../fwdtest.RData")`

Step 2: 'mars.control' Object

- Create followings for the 'mars.control' class using the information provided in mars5.pdf,

- ▶ **Helper:**

- ▶ Use default values `Mmax=2`, `d=3`, `trace=FALSE`.
- ▶ `Mmax` should be an even integer ≥ 2 . If not, coerce user's input and throw a warning.

- ▶ **Validator:**

- ▶ Check that `Mmax` is an even integer ≥ 2 , `d` is numeric, and `trace` is logical.

- ▶ **Constructor:**

- ▶ Its input is a list, its output is an object of class `mars.control`.

- **Constructor:** `new_mars.control()`
- **Validator:** `validate_mars.control()`
- **Helper:** `mars.control()`

- Use your helper with no arguments, so that it uses all defaults, to create a `mars.control` object named 'mc'.

- `mc <- mars.control()`

- Use `all.equal(mc, mctest)` to verify that `mc` is the same as `mctest`.
 - `all.equal(mc, mctest)` # should return 'TRUE'

Step 3: 'model.matrix'

- Execute the following lines of R code,

```
set.seed(123); n <- 10  
data <- data.frame(x1=rnorm(n),x2=rnorm(n),y=rnorm(n))  
formula <- formula(y ~.)
```

- Use `model.matrix()` to extract the matrix of predictors from the input formula and data frame (refer to mars4.pdf).

```
mars <- function(formula,data,control=NULL...) {  
  cc <- match.call() # save the call  
  mf <- model.frame(formula,data)  
  y <- model.response(mf)  
  mt <- attr(mf, "terms")  
  x <- model.matrix(mt, mf)  
  fwd <- fwd_stepwise(y,x,control) # Note change from ear  
  bwd <- bwd_stepwise(fwd,control)  
  # Now prepare the output and return a "mars" object --  
  # to be discussed in a future lecture  
}
```

- Step through the code snippet to create the matrix `x` of predictors.
- Use `head()` to print the first few rows. The first column is a column of 1's for the intercept and is not a predictor.
- Remove this column from your `x`.
- Use `all.equal(x,xtest)` to test that the modified `x` is the same as `xtest`.

Algorithm 1 \rightarrow Algorithm 2

Modifications to MARS -Algorithm 1, including:

- ▶ Replace the indicator functions $H()$ with mirror-image hinge functions $h()$.
- ▶ Do not remove parent basis function $B_m(x)$ after it is split. We are therefore adding **pairs** of basis functions in each outer-loop iteration.
- ▶ Only split basis function $B_m(x)$ into $B_m(x)h(t - x_v)$ and $B_m(x)h(x_v - t)$ for variables x_v not already involved in B_m .

Refer to Section 3.3 of the MARS manuscript for more details.

Hinge Function $h()$

Some properties of the desired hinge function $h()$ (refer to mars5.pdf, p.6),

- Takes 3 inputs: s , $x[v]$ and t .
- Returns $\max(0, x_v - t)$ or $\max(0, t - x_v)$, depending on the value of s .
- Can use `pmax()`.
- No longer an indicator function.

Step 4: Updating Bfuncs and B (mars5.pdf)

- 'Mmax' is the maximum number of non-intercept basis functions.
- The list Bfuncs is of length Mmax+1.
- Element 'm' of Bfuncs is a data frame with columns s, v and t, and as many rows as there are hinge functions that make up Bm.
- The first element of Bfuncs is for the constant (intercept) basis function that is not made of any hinge functions, so we will leave it empty.
- Implement followings,
 - Initialize B with your init_B() function and Bfuncs to be an empty list of length mc\$Mmax+1.
 - Change the way B and Bfuncs are updated within the loops over m, v and t, so that we add pairs of basis functions.
 - Previously we add one child basis function and replace the parent basis function with the other child basis function
 - Replace the step function H() with the hinge function h().

Step 5: Changes to Loops (mars5.pdf)

- In `recpart_fwd()` function, there are four nested loops,
 - the number of basis functions to construct (M loop)*
 - the basis function to split (m loop)
 - the variable to split on (v loop)*
 - the split point (t loop).
- Implement followings,
 - Replace the M loop with a loop over pairs *i*, and set the value of M from the value of *i*.
 - In the v loop, `recpart_fwd()` was allowed to split on any of the *n* explanatory variables, but `fwd_stepwise()` is restricted to split on variables that are not already in basis function *m*. For a given *m*,
 - Write the `for()` statement that loops over variables *v* not already in `Bfuncs[[m]]`.
 - Hints:
 - (1) What is in `Bfuncs[[m]][,"v"]`?
 - (2) How might the `setdiff()` function be useful?

Step 6: Test the Output of fwd_stepwise()

- Return the list 'list(y=y, B=B, Bfuncs=Bfuncs)' from fwd_stepwise();
- Before you return, set the colnames of B with:
`colnames(B) <- paste0("B",0:(ncol(B)-1)))`
- Use the y, x, and mc from steps 2 and 3 as the input to your fwd_stepwise() function;
- Save the output as fwd;
- Check that your fwd is the same as fwdtest with all.equal(fwd, fwdtest).

```
fwd <- fwd_stepwise(y, x, mc)  
all.equal(fwd, fwdtest) # should return 'TRUE'
```

Quiz Date Update

The last two lab quizzes has been rescheduled. The new dates are:

- Lab Quiz 4: March 23/24
- Lab Quiz 5: April 6/7