

# Statistics 360: Advanced R for Data Science

## MARS, part V

Becky Lin



## More details on the implementation

- ▶ So far we have discussed the formula and data inputs, and pre-processing.
- ▶ Today we'll talk about the `mars.control` object input, some hints about the forward algorithm, and packaging the output as a `mars` object.
- ▶ You will implement the forward and backward algorithms yourself.

## `mars.control` object as input

- ▶ From the mars4 lecture we noted that `mars()` should take an object of class `mars.control` as input.
- ▶ The elements of a `mars.control` object are `Mmax`, `d` and `trace`.
- ▶ Write a constructor, validator and helper for this class.
- ▶ Helper:
  - ▶ Use default values `Mmax=2`, `d=3`, `trace=FALSE`.
  - ▶ `Mmax` should be an even integer  $\geq 2$ . If not, coerce user's input and throw a warning.
- ▶ Validator:
  - ▶ Check that `Mmax` is an even integer  $\geq 2$ , `d` is numeric, and `trace` is logical.
- ▶ Constructor:
  - ▶ Its input is a list, its output is an object of class `mars.control`.

## Hints on fwd\_stepwise()

- ▶ You should be working on a first draft of the `fwd_stepwise()` function, as a modification of your `recpart_fwd()` algorithm from lab 5.
- ▶ The key modifications are
  - ▶ 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  $B_m$ .

## Indicators to hinge functions

- ▶ This modification is relatively straightforward.
- ▶ Your hinge function  $h()$  should take arguments  $s$  (sign),  $x$  (variable) and  $t$  (knot) and return the maximum of 0 and  $s*(x-t)$  for each value of  $x$ .
- ▶ You may find the `pmax()` function useful.

## Adding pairs of basis functions

- ▶ Can **initialize B** as in `recpart_fwd()`, but we'll initialize **Bfuncs** to be of length  $M_{\max}+1$  to match B.
  - ▶ `Bfuncs <- vector(mode="list",length=Mmax+1)`
  - ▶ Recall: B has  $M_{\max}+1$  columns, including the intercept  $B_0$ .
- ▶ Replace the for loop over basis functions M from 1 to  $M_{\max}$  with a loop over **pairs 1 to  $M_{\max}/2$** .
- ▶ When adding pair  $i$ , there are currently  $M=2*i-1$  basis functions to consider splitting.
  - ▶ E.G., when you add pair 1 there is  $2*1-1=1$  (the constant basis function), when you add pair 2 there are currently  $2*2-1=3$  (the constant and the first pair), etc.
- ▶ As in `recpart_fwd()` when you loop over basis functions, variables and split points, keep track of **lof\_best** and the best **m, v and t**. When you finish the three loops, construct left- and right-split matrices from `Bfuncs[[m]]` and add this **pair** to the B matrix and the Bfuncs list.
  - ▶ The indices of pair  $i$  in B and Bfuncs are  $2*i$  and  $2*i+1$ .

## Restriction on splitting

- ▶ Only split basis function  $B_m(x)$  into  $B_m(x)h(+(x_v - t))$  and  $B_m(x)h(-(x_v - t))$  for variables  $x_v$  not already involved in  $B_m$ .
- ▶ You will need to consult the `Bfuncs[m]` to see which basis functions make up  $B_m$ .



## Value/output

- ▶ object of S3 class `mars`.
- ▶ inherits from class `lm` and includes all of the components of the `lm()` from the final fit
  - ▶ Use `c()` to combine these with any of your own components.
  - ▶ Note: `bwd_selection()` will *select* the best model but will not *return* the fit. You will need to call `lm()` after `bwd_selection()` to obtain the final fit.
- ▶ include `Bfuncs` data structure from final fit.
- ▶ write a constructor for class `mars` – no need for a validator or helper since you are the only one who will call the constructor.

# Methods

- ▶ Use `methods()` to find a list of methods implemented for the S3 class `lm`.
- ▶ Write more informative `print` and `summary` methods for `lm` objects
- ▶ Write a `plot` method.
  - ▶ The details are up to you, but you should consult Section 3.5 of the Friedman paper (ANOVA decomposition).
  - ▶ Two sources of inspiration are the `plot.earth` method for `earth` objects (see the `earth` package), and `plot.Gam` for plotting generalized additive model components (see the `gam` package).
- ▶ Write a `predict` method with the same interface as `predict.lm`.
- ▶ You can use the `residuals()`, `fitted()`, `hatvalues()` and other methods for `lm` objects for methods that depend only on the final `lm`.