# Multivariate Adaptive Regression Splines (MARS)

## Github Page

https://github.com/DustinPoon/STAT360Project

## Description

Using methods from Friedman's paper "Multivariate Adaptive Regression Splines" to construct a regression model.

## Usage

```
mars(formula, data, control = NULL)
```

## Arguments

*formula* an R formula.

*data* a data frame containing the data for the model.

*control* an object of class 'mars.control'.

## Details

Jerome H. Friedman introduced a type of regression analysis called Multivariate Adaptive Regression Spline (MARS). MARS is a non-parametric regression technique that automatically models nonlinearities and interactions between variables. MARS begins by recursively partitioning the data into two subsets and fitting a linear regression model to the regions. The predictor variable is where the subsetting occurs and is chosen by the forward stepwise method, which selects the best predictor variable. Again, repeated for each subset. A new predictor variable is added to the model, using the residuals from the two models. The process is repeated until the Mmax of basis functions is reached. The resulting model consists of the selected basis functions and their formulas, the names of the predictor variables, and the fitted linear model object.

## Key Aspects in Mars Algorithm

### Forward Step-wise

The forward step-wise algorithm recursively partitions the predictor variable space into sub-regions by adding pairs of basis functions, each representing a sub-region. The algorithm selects the optimal basis function to split on by iterating over four loops, with each iteration selecting a variable and split point based on the GCV criterion. The basis functions are recorded in a list called Bfuncs and a matrix called B.

### Backward Step-wise

The backward step-wise algorithm removes terms one by one, selecting the least effective term at each step until it finds the best sub-model. Model subsets are compared using the GCV criterion, implemented using two for loops. The outer loop iterates over all possible model sizes, while the inner loop iterates over all model terms in the current model and tries removing one basis function at a time. The final model is selected based on the lowest observed lack of fit value using the GCV criterion.

### Hinge Function

MARS models use hinge functions, which are constructed using a variable x and a split-point (knot) “a.” The hinge function takes the form max(0, x - a) or max(0, a - x) and can be used to partition data into disjoint regions. These regions can be treated independently and the hinge functions can be multiplied together to form non-linear functions.

### Generalized Cross Validation (GCV) Criterion

When selecting models in the MARS algorithm, the Generalized Cross Validation (GCV) criterion is used to measure “lack of fit” (LOF) instead of the Residual Sum of Squares (RSS). The RSS tends to favor larger models because it decreases as we add predictors. Cross-Validation is a better approach but can be time-consuming, which is why we use the GCV criterion as an approximation of Cross-Validation. The GCV criterion can be calculated using the following formula:

RSS×(N/(N - C’(M))ˆ2) N is the number of rows in the dataset. M is one less than the number of coefficients in the fitted model. C’(M) is the sum of the hat-values from the fitted model and the smoothing parameter d times M, where d is typically set to 3, according to Friedman.

# Value

An S3 model of class ‘mars’.

# Author(s)

Group DJT: Dustin Poon, James Lee, Tyler Oh

# References

Jerome H. Friedman, The Annals of Statistics, Mar., 1991, Vol. 19, No. 1 (Mar., 1991), pp. 1-67.

## See Also

*mars.control* for constructing control object.

*anova.mars* for the ANOVA table.

*plot.mars* for plotting the fitted basis function.

*predict.mars* for Predicting from the fitted model.

*print.mars* for printing out a mars object.

*summary.mars* for more detailed summaries of mars object.

*For more detailed explanations, please type ?function_name in your R or Rstudio console. (i.e., ?mars)*

## Examples

### Data

#### Wage: R built-in dataset

#### iris: The dataset from library(ISLR)

#### CarPrice: This dataset is extracted from https://www.kaggle.com/datasets/hellbuoy/car-price-prediction

#### Setup

```
library(Mars)
CarPrice <- read.csv("/Users/tyler/Mars/vignettes/CarPrice_Assignment.csv")
```

#### Control

```
mc <- mars.control(Mmax=10)
```

#### Wage data

```
fit.Wage <- mars(wage ~ age + education, ISLR::Wage, mc)
print.mars(fit.Wage)
summary.mars(fit.Wage)
predict.mars(fit.Wage, newdata = data.frame(age=ISLR::Wage$age,
                                            education = ISLR::Wage$education))
anova.mars(fit.Wage)
plot.mars(fit.Wage)
```

#### iris Dataset

```
fit.iris <- mars(Sepal.Length ~., iris, mc)
print.mars(fit.iris)
summary.mars(fit.iris)
predict.mars(fit.iris, newdata = data.frame(sepal.width = iris$Sepal.Width,
                                            petal.length = iris$Petal.Length,
                                            petal.width = iris$Petal.Width,
                                            Species = iris$Species))
anova.mars(fit.iris)
plot.mars(fit.iris)
```

### CarPrice dataset

```
fit.CarPrice <- mars(price ~ citympg + highwaympg + horsepower + fueltype, CarPrice, mc)
print.mars(fit.CarPrice)
summary.mars(fit.CarPrice)
predict.mars(fit.CarPrice, newdata = data.frame(citympg = CarPrice$citympg,
                                                highwaympg = CarPrice$highwaympg,
                                                horsepower = CarPrice$highwaympg,
                                                fueltype = CarPrice$highwaympg))
anova.mars(fit.CarPrice)
plot.mars(fit.CarPrice)
```