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)
anova.mars(object, ...)
plot.mars(x, ...)
print.mars(x, ...)
predict.mars(object, newdata, ...)
summary.mars(object, ...)
```

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.

Setup

```
library(Mars)
options(max.print=100)
```

Examples

Data

Wage: R built-in dataset

iris: The dataset from library(ISLR)

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

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

Control

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

Example with Wage Dataset

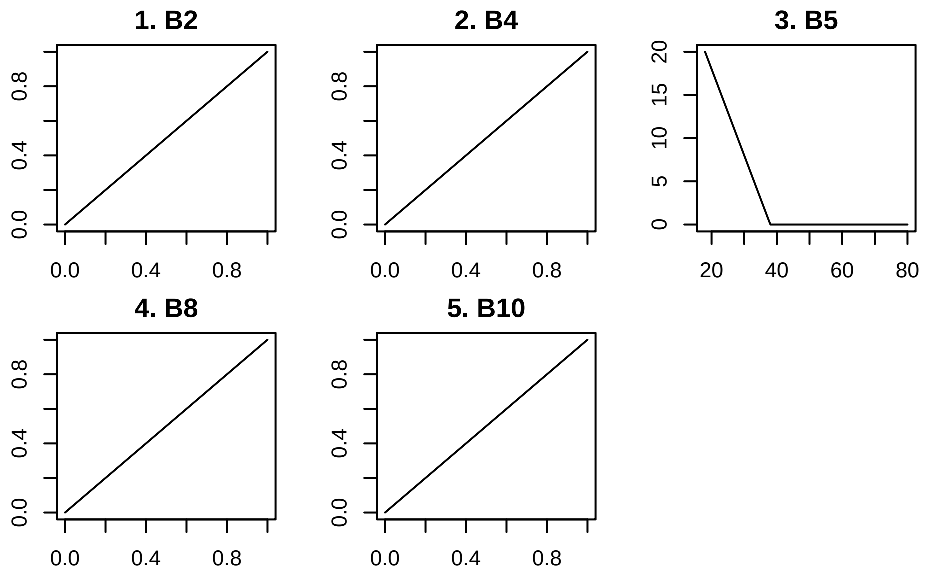
```
fit.Wage <- mars(wage ~ age + education, ISLR::Wage, mc)
```

```
print.mars(fit.Wage)
#>
#>  Call:
#> mars(formula = wage ~ age + education, data = ISLR::Wage, control = mc)
#>
#>  Coefficients:
#>        B0        B2        B4        B5        B8       B10
#> 90.813594 62.942867 38.542884 -1.982426 23.788965 11.151822
```

```
predict.mars(fit.Wage, newdata = data.frame(age=ISLR::Wage$age,
                                             education = ISLR::Wage$education))
#>   [1]  51.16508 101.60252 114.60256 129.35648 101.96542 129.35648 114.60256
#>   [8]  98.74315 114.60256 101.96542 114.60256  94.03571  96.01814 129.35648
#>  [15] 101.96542 114.60256 127.37405 153.75646 129.35648 112.62013 129.35648
#>  [22] 101.96542 129.35648 129.35648 101.96542 153.75646 101.96542  94.03571
#>  [29] 101.96542  62.31690 153.75646 114.60256 143.84433  94.03571 125.39163
#>  [36] 153.75646 129.35648 103.58494 102.70801  69.00691 109.53222 107.54980
#>  [43]  90.81359  90.81359 101.96542 153.75646 123.40920 153.75646  65.04206
#>  [50]  92.05329 153.75646 114.60256 129.35648  84.86617 137.89706  70.24660
#>  [57] 153.75646 109.53222 101.96542  94.03571 129.35648 114.60256 101.96542
#>  [64] 114.60256 153.75646 114.60256 101.96542 127.37405 101.96542  68.26418
#>  [71] 101.96542  88.08844  76.19388 102.70801  90.81359 153.75646 129.35648
#>  [78]  61.07721 101.96542  90.81359 153.75646 114.60256 153.75646  63.05963
#>  [85] 101.96542  76.19388  94.03571  90.81359 129.35648 153.75646 129.35648
#>  [92] 101.96542  80.90147 145.82676  70.24660 101.96542 114.60256 129.35648
#>  [99] 129.35648 119.44435
#>  [ reached getOption("max.print") -- omitted 2900 entries ]
```

```
summary.mars(fit.Wage)
#>
#> B0: Intercept
#>
#> B2: Sign: 1
#> Split Variables: 5
#> Split Points: 0
#>
#> B4: Sign: 1
#> Split Variables: 4
#> Split Points: 0
#>
#> B5: Sign: -1
#> Split Variables: 1
#> Split Points: 38
#>
#> B8: Sign: 1
#> Split Variables: 3
#> Split Points: 0
#>
#> B10: Sign: 1
#> Split Variables: 2
#> Split Points: 0
#>
#> Call:
#> mars(formula = wage ~ age + education, data = ISLR::Wage, control = mc)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -115.151  -19.758   -2.882   14.156  216.377
#>
#> Coefficients:
#>     Estimate Std. Error t value Pr(>|t|)
#> B0    90.814      2.201  41.254  < 2e-16 ***
#> B2    62.943      2.762  22.790  < 2e-16 ***
#> B4    38.543      2.544  15.152  < 2e-16 ***
#> B5    -1.982      0.134 -14.799  < 2e-16 ***
#> B8    23.789      2.560   9.293  < 2e-16 ***
#> B10   11.152      2.434   4.582 4.78e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 35.26 on 2994 degrees of freedom
#> Multiple R-squared:  0.9127, Adjusted R-squared:  0.9125
#> F-statistic:  5218 on 6 and 2994 DF,  p-value: < 2.2e-16
```

```
plot.mars(fit.Wage)
```



```
anova.mars(fit.Wage)
#> Analysis of Variance Table
#>
#> Response: y
#>             Df  Sum Sq Mean Sq F value    Pr(>F)
#> B2           1  763499  763499 613.938 < 2.2e-16 ***
#> B4           1  344600  344600 277.097 < 2.2e-16 ***
#> B5           1  267963  267963 215.472 < 2.2e-16 ***
#> B8           1   96544   96544  77.632 < 2.2e-16 ***
#> B10          1   26114   26114  20.999 4.783e-06 ***
#> Residuals 2994 3723365    1244
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example with Iris Dataset

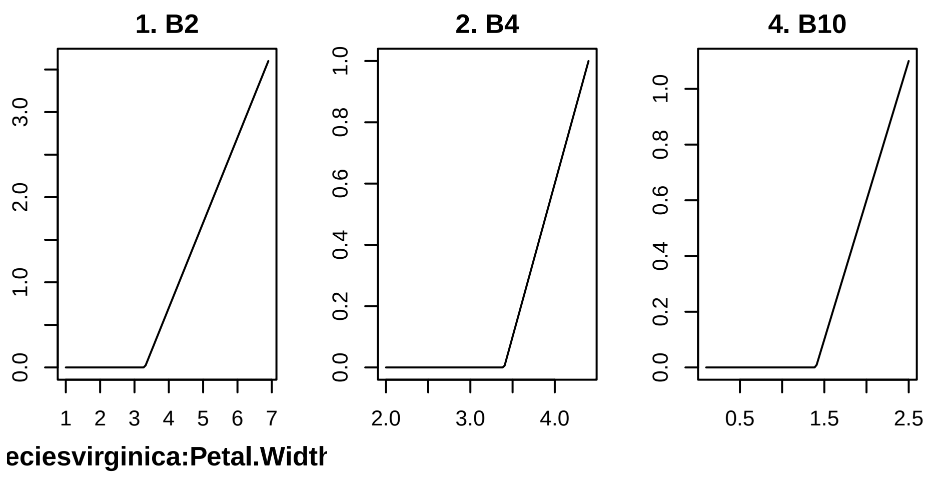
```
fit.iris <- mars(Sepal.Length ~., iris, mc)
```

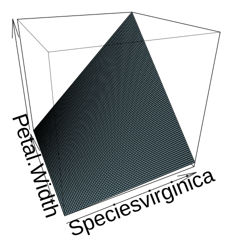
```
print.mars(fit.iris)
#>
#>  Call:
#> mars(formula = Sepal.Length ~ ., data = iris, control = mc)
#>
#>  Coefficients:
#>         B0         B2         B4         B7        B10
#>  4.9254642  1.0451176  0.7902179 -0.8034245 -0.7649694
```

```
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))
#>   [1] 5.004486 4.925464 4.925464 4.925464 5.083508 5.320573 4.925464 4.925464
#>   [9] 4.925464 4.925464 5.162530 4.925464 4.925464 4.925464 5.399595 5.715682
#>  [17] 5.320573 5.004486 5.241551 5.241551 4.925464 5.162530 5.083508 4.925464
#>  [25] 4.925464 4.925464 4.925464 5.004486 4.925464 4.925464 4.925464 4.925464
#>  [33] 5.478617 5.557639 4.925464 4.925464 5.004486 5.083508 4.925464 4.925464
#>  [41] 5.004486 4.925464 4.925464 5.004486 5.241551 4.925464 5.241551 4.925464
#>  [49] 5.162530 4.925464 6.388629 6.103108 6.521155 5.657047 6.207620 6.179605
#>  [57] 6.235635 4.925464 6.284117 5.552535 5.134488 5.789573 5.657047 6.388629
#>  [65] 5.238999 6.075094 6.103108 5.761558 6.103108 5.552535 6.187153 5.657047
#>  [73] 6.521155 6.388629 5.970582 6.075094 6.493141 6.472673 6.103108 5.134488
#>  [81] 5.448023 5.343511 5.552535 6.653682 6.103108 6.026611 6.312132 6.075094
#>  [89] 5.761558 5.657047 6.075094 6.284117 5.657047 4.925464 5.866070 5.866070
#>  [97] 5.866070 5.970582 4.925464 5.761558
#>  [ reached getOption("max.print") -- omitted 50 entries ]
```

```
summary.mars(fit.iris)
#>
#> B0: Intercept
#>
#> B2: Sign: 1
#> Split Variables: 2
#> Split Points: 3.3
#>
#> B4: Sign: 1
#> Split Variables: 1
#> Split Points: 3.4
#>
#> B7: Sign: 1
#> Split Variables: 5
#> Split Points: 0
#> Sign: -1
#> Split Variables: 3
#> Split Points: 2.3
#>
#> B10: Sign: 1
#> Split Variables: 3
#> Split Points: 1.4
#>
#> Call:
#> mars(formula = Sepal.Length ~ ., data = iris, control = mc)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.70311 -0.22439 -0.01008  0.21406  0.78180
#>
#> Coefficients:
#>     Estimate Std. Error t value Pr(>|t|)
#> B0   4.92546    0.04158 118.443  < 2e-16 ***
#> B2   1.04512    0.05002  20.893  < 2e-16 ***
#> B4   0.79022    0.16229   4.869 2.90e-06 ***
#> B7  -0.80342    0.15324  -5.243 5.48e-07 ***
#> B10 -0.76497    0.13532  -5.653 8.10e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.3091 on 145 degrees of freedom
#> Multiple R-squared:  0.9973, Adjusted R-squared:  0.9973
#> F-statistic: 1.09e+04 on 5 and 145 DF,  p-value: < 2.2e-16
```

```
plot.mars(fit.iris)
```





```
anova.mars(fit.iris)
#> Analysis of Variance Table
#>
#> Response: y
#>            Df Sum Sq Mean Sq F value    Pr(>F)
#> B2          1 81.988  81.988 857.889 < 2.2e-16 ***
#> B4          1  1.858   1.858  19.441 2.008e-05 ***
#> B7          1  1.410   1.410  14.756 0.0001824 ***
#> B10         1  3.054   3.054  31.958 8.096e-08 ***
#> Residuals 145 13.858   0.096
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example with CarPrice dataset

```
fit.CarPrice <- mars(price ~ citympg + highwaympg + horsepower + fueltype, CarPrice, mc)
```

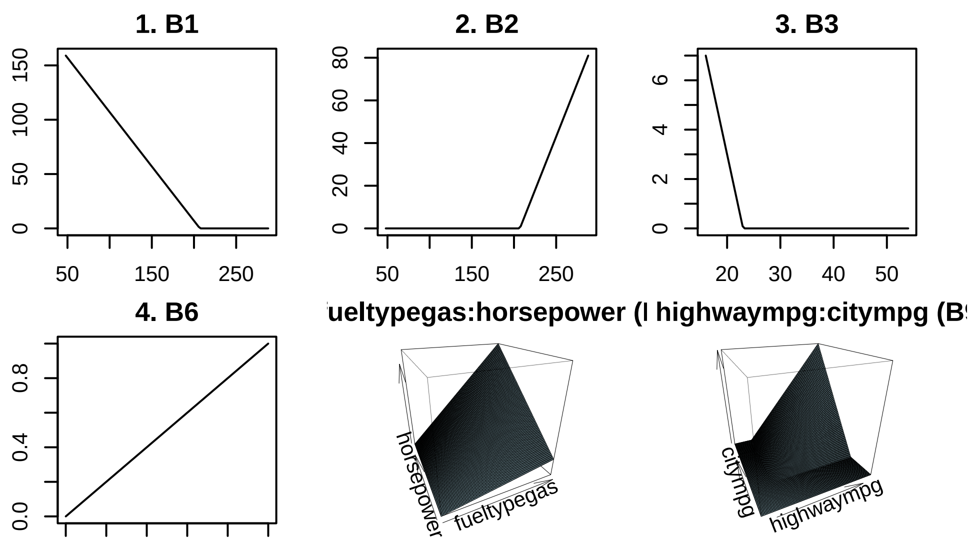
```
print.mars(fit.CarPrice)
#>
#>  Call:
#> mars(formula = price ~ citympg + highwaympg + horsepower + fueltype,
#>     data = CarPrice, control = mc)
#>
```

```
#>  Coefficients:
#>          B0          B1          B2          B3          B6          B7
#>  49517.3112   -276.3302   -167.0311   3335.4630 -20845.1130    185.0028
#>          B9
#>    462.7486
```

```
predict.mars(fit.CarPrice, newdata = data.frame(citympg = CarPrice$citympg,
                                                highwaympg = CarPrice$highwaympg,
                                                horsepower = CarPrice$highwaympg,
                                                fueltype = CarPrice$highwaympg))
#>   [1] 110001.776 110001.776 112441.256 102314.297 108877.624 110808.742
#>   [7] 110808.742 110808.742 112655.460 108877.624 104012.799 104012.799
#>  [13] 109968.782 109968.782 109883.245 108877.624 108877.624 112655.460
#>  [19] -38872.772  46563.268  46563.268  59210.410  75406.082 102314.297
#>  [25]  75406.082  75406.082  75406.082 102314.297 102314.297 108806.222
#>  [31] -49451.406  75406.082  53071.842  91820.233  91820.233  91820.233
#>  [37]  91820.233  94998.758  94998.758  94998.758  94998.758 105341.296
#>  [43] 100245.790 104012.799  46563.268  46563.268 104012.799 113989.370
#>  [49] 113989.370 115547.174 100245.790  75406.082  75406.082  75406.082
#>  [55]  75406.082 106433.697 106433.697 106433.697 106433.697  97807.276
#>  [61]  97807.276  97807.276  97807.276  53071.842  97807.276 113703.765
#>  [67]  70377.530 108032.250 108032.250 108032.250 108032.250 114953.275
#>  [73] 114953.275 115771.068 115771.068 108806.222  59210.410  75406.082
#>  [79]  75406.082 102314.297 102314.297  97807.276 108806.222 108806.222
#>  [85] 108806.222  97807.276  97807.276 102314.297 102314.297  80064.628
#>  [91]  -9356.902  80064.628  80064.628  80064.628  80064.628  80064.628
#>  [97]  80064.628  80064.628  80064.628  91820.233
#>  [ reached getOption("max.print") -- omitted 105 entries ]
```

```
summary.mars(fit.CarPrice)
#>
#> B0: Intercept
#>
#> B1: Sign: -1
#> Split Variables: 3
#> Split Points: 207
#>
#> B2: Sign: 1
#> Split Variables: 3
#> Split Points: 207
#>
#> B3: Sign: -1
#> Split Variables: 2
#> Split Points: 23
#>
#> B6: Sign: 1
#> Split Variables: 4
#> Split Points: 0
#>
#> B7: Sign: 1
#> Split Variables: 4
#> Split Points: 0
#> Sign: -1
#> Split Variables: 3
#> Split Points: 161
#>
#> B9: Sign: 1
#> Split Variables: 2
#> Split Points: 23
#> Sign: -1
#> Split Variables: 1
#> Split Points: 23
#>
#> Call:
#> mars(formula = price ~ citympg + highwaympg + horsepower + fueltype,
#>     data = CarPrice, control = mc)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -7038.9 -1481.8  -420.8  1088.9 16215.6
#>
#> Coefficients:
#>     Estimate Std. Error t value Pr(>|t|)
#> B0  49517.31    2476.19  19.997  < 2e-16 ***
#> B1   -276.33      19.52 -14.155  < 2e-16 ***
#> B2   -167.03      34.08  -4.901 1.98e-06 ***
#> B3   3335.46     236.87  14.081  < 2e-16 ***
#> B6 -20845.11    1805.39 -11.546  < 2e-16 ***
#> B7    185.00      22.30   8.295 1.65e-14 ***
#> B9    462.75      68.02   6.803 1.18e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2802 on 198 degrees of freedom
#> Multiple R-squared:  0.9684, Adjusted R-squared:  0.9672
#> F-statistic: 865.9 on 7 and 198 DF,  p-value: < 2.2e-16
```

```
plot.mars(fit.CarPrice)
```



```
anova.mars(fit.CarPrice)
#> Analysis of Variance Table
#>
#> Response: y
#>            Df     Sum Sq    Mean Sq   F value    Pr(>F)
#> B1          1 8661205571 8661205571 1102.9291 < 2.2e-16 ***
#> B2          1    6938112    6938112    0.8835    0.3484
#> B3          1 1184689849 1184689849  150.8599 < 2.2e-16 ***
#> B6          1  799368847  799368847  101.7927 < 2.2e-16 ***
#> B7          1  449070291  449070291   57.1852 1.451e-12 ***
#> B9          1  363490003  363490003   46.2873 1.183e-10 ***
#> Residuals 198 1554876690    7852913
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```