Regression Tree and Multivariate Adaptive Regression Splines (MARS)

Munmun Biswas

Dept. of Statistics, Brahmananda Keshab Chandra College

July 28, 2020

Classification Problem

- CART in classification problem

Classification Problem

- CART in classification problem
  - We have discussed the idea

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm
- To improve the effectivity certain ensemble methods are proposed

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm
- To improve the effectivity certain ensemble methods are proposed
  - Bagging

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm
- To improve the effectivity certain ensemble methods are proposed
  - Bagging
  - Boosting

Classification Problem

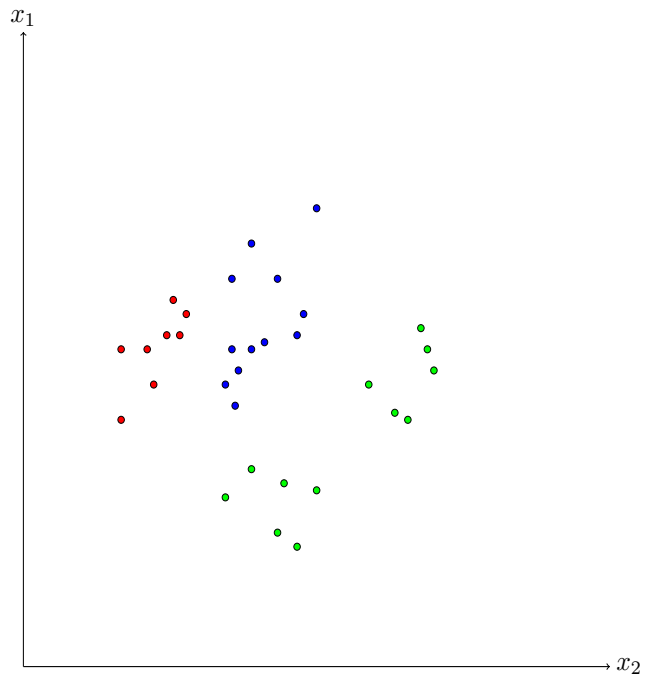
- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm
- To improve the effectivity certain ensemble methods are proposed
  - Bagging
  - Boosting
  - Random Forest

Classification Problem

- CART in classification problem
  - We have discussed the idea
  - We have seen the rpart package for implementation
  - Also certain limitations of CART algorithm
- To improve the effectivity certain ensemble methods are proposed
  - Bagging
  - Boosting
  - Random Forest
- To be discussed in next class

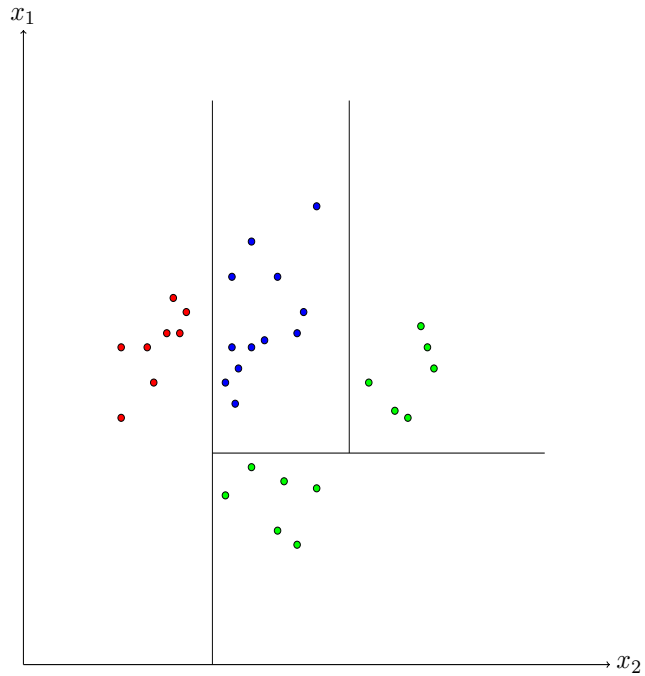
Regression Preblem

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where
  $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$



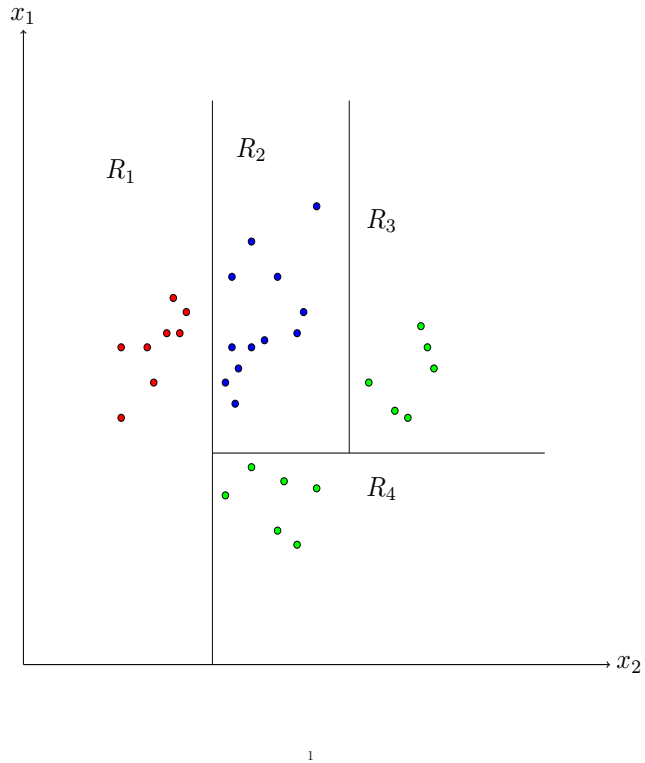
Regression Problem

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- The CART algorithm splits the x-space into partitions, say $\{R_1, R_2, \ldots, R_M\}$



Regression Problem

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- The algorithm splits the x-space into partitions, say $\{R_1, R_2, \ldots, R_M\}$
- The regression tree model of response as $f(x) = \sum_{m=1}^{M} c_m \mathbf{I}(x \in R_m)$
- $\hat{c}_m = ave(y_i | \mathbf{x}_i \in R_m)$



The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.
- The model begins with the entire data set, S. It searches every distinct value of every input variable to find the predictor and split value, that partitions the data into two regions R_1 and R_2 such that the overall sums of squares error are minimized.

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.
- The model begins with the entire data set, S. It searches every distinct value of every input variable to find the predictor and split value, that partitions the data into two regions R_1 and R_2 such that the overall sums of squares error are minimized.
- Minimize $\{SSE = \sum_{i \in R_1}(y_i - c_1)^2 + \sum_{i \in R_2}(y_i - c_2)^2\}$

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.
- The model begins with the entire data set, S. It searches every distinct value of every input variable to find the predictor and split value, that partitions the data into two regions R_1 and R_2 such that the overall sums of squares error are minimized.
- Minimize $\{SSE = \sum_{i \in R_1}(y_i - c_1)^2 + \sum_{i \in R_2}(y_i - c_2)^2\}$
- Having found the best binary split, we partition the data into the two resulting regions.

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.
- The model begins with the entire data set, S. It searches every distinct value of every input variable to find the predictor and split value, that partitions the data into two regions R_1 and R_2 such that the overall sums of squares error are minimized.
- Minimize $\{SSE = \sum_{i \in R_1}(y_i - c_1)^2 + \sum_{i \in R_2}(y_i - c_2)^2\}$
- Having found the best binary split, we partition the data into the two resulting regions.
- Repeat the splitting process on each of the two regions.

The CART algorithm

- The partitioning of variables are done in a top-down, greedy fashion.
- A partition performed earlier in the tree will not change based on later partitions.
- The model begins with the entire data set, S. It searches every distinct value of every input variable to find the predictor and split value, that partitions the data into two regions R_1 and R_2 such that the overall sums of squares error are minimized.
- Minimize $\{SSE = \sum_{i \in R_1}(y_i - c_1)^2 + \sum_{i \in R_2}(y_i - c_2)^2\}$
- Having found the best binary split, we partition the data into the two resulting regions.
- Repeat the splitting process on each of the two regions.
- This process is continued until some stopping criterion is reached.

Cost Complexity Parameter

- Stopping rule

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.
  - **maxdepth**: the maximum number of internal nodes between the root node and the terminal nodes.

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.
  - **maxdepth**: the maximum number of internal nodes between the root node and the terminal nodes.
- We typically grow a very large tree as defined in the previous section and then prune it back to find an optimal subtree.

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.
  - **maxdepth**: the maximum number of internal nodes between the root node and the terminal nodes.
- We typically grow a very large tree as defined in the previous section and then prune it back to find an optimal subtree.
- There is often a balance to be achieved in the depth and complexity of the tree to optimize predictive performance on some unseen data.

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.
  - **maxdepth**: the maximum number of internal nodes between the root node and the terminal nodes.
- We typically grow a very large tree as defined in the previous section and then prune it back to find an optimal subtree.
- There is often a balance to be achieved in the depth and complexity of the tree to optimize predictive performance on some unseen data.
- minimize$\{SSE + \alpha \times |T|\}$ where $|T|$ is the number of terminal nodes of the tree.

Cost Complexity Parameter

- Stopping rule
  - **minsplit**: the minimum number of data points required to attempt a split before it is forced to create a terminal node.
  - **maxdepth**: the maximum number of internal nodes between the root node and the terminal nodes.
- We typically grow a very large tree as defined in the previous section and then prune it back to find an optimal subtree.
- There is often a balance to be achieved in the depth and complexity of the tree to optimize predictive performance on some unseen data.
- minimize$\{SSE + \alpha \times |T|\}$ where $|T|$ is the number of terminal nodes of the tree.
- Obtain the smallest pruned tree that has the lowest penalized error.

Implementation of Regression Tree

- R package *rpart*

Implementation of Regression Tree

- R package *rpart*
- Using *rattle()*

Implementation of Regression Tree

- R package *rpart*
- Using *rattle()*
  - Rattle is a free graphical user interface for Data Science, developed using R. R is a free software environment for statistical computing, graphics, machine learning and artificial intelligence. Together Rattle and R provide a sophisticated environment for data science, statistical analyses, and data visualisation.

Data Description

Body girth measurements and skeletal diameter measurements, as well as age, weight, height and gender, are given for 507 physically active individuals- 247 men and 260 women.

	Type	Column	Name of the Variable
2			Age (years)
3			Weight (kg)
4			Height (cm)
5			Gender (1 - male, 0 - female)
6	Skeletal Measurements	VAR1	Biacromial diameter (see Fig. 2)
7		VAR2	Biiliac diameter, or "pelvic breadth" (see Fig. 2)
8		VAR3	Bitrochanteric diameter (see Fig. 2)
9		VAR4	Chest depth between spine and sternum at nipple level,mid-expiration
10	Girth Measurements	VAR5	Chest diameter at nipple level, mid-expiration
11		VAR6	Elbow diameter, sum of two elbows
12		VAR7	Wrist diameter, sum of two wrists
13		VAR8	Knee diameter, sum of two knees
14		VAR9	Ankle diameter, sum of two ankles
15		VAR10	Shoulder girth over deltoid muscles
16		VAR11	Chest girth, nipple line in males and just above breast tissue in females, mid-expiration
17		VAR12	Waist girth, narrowest part of torso below the rib cage, average of contracted and relaxed position
18		VAR13	Navel (or "Abdominal") girth at umbilicus and iliac crest, iliac crest as a landmark
19		VAR14	Hip girth at level of bitrochanteric diameter
20		VAR15	Thigh girth below gluteal fold, average of right and left girths
21		VAR16	Bicep girth, flexed, average of right and left girths
22		VAR17	Forearm girth, extended, palm up, average of right and left girths
23		VAR18	Knee girth over patella, slightly flexed position, average of right and left girths
24		VAR19	Calf maximum girth, average of right and left girths
25		VAR20	Ankle minimum girth, average of right and left girths
26		VAR21	Wrist minimum girth, average of right and left girths
27	Other		
28			
29			

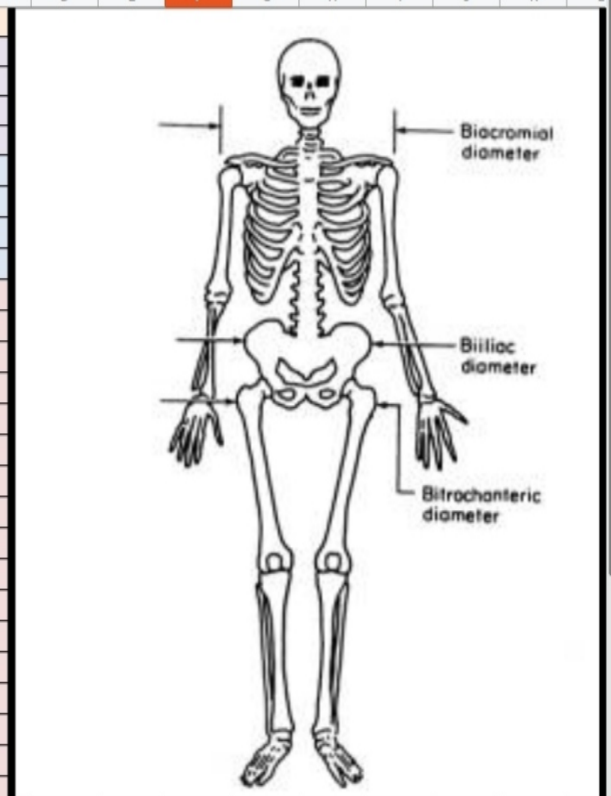
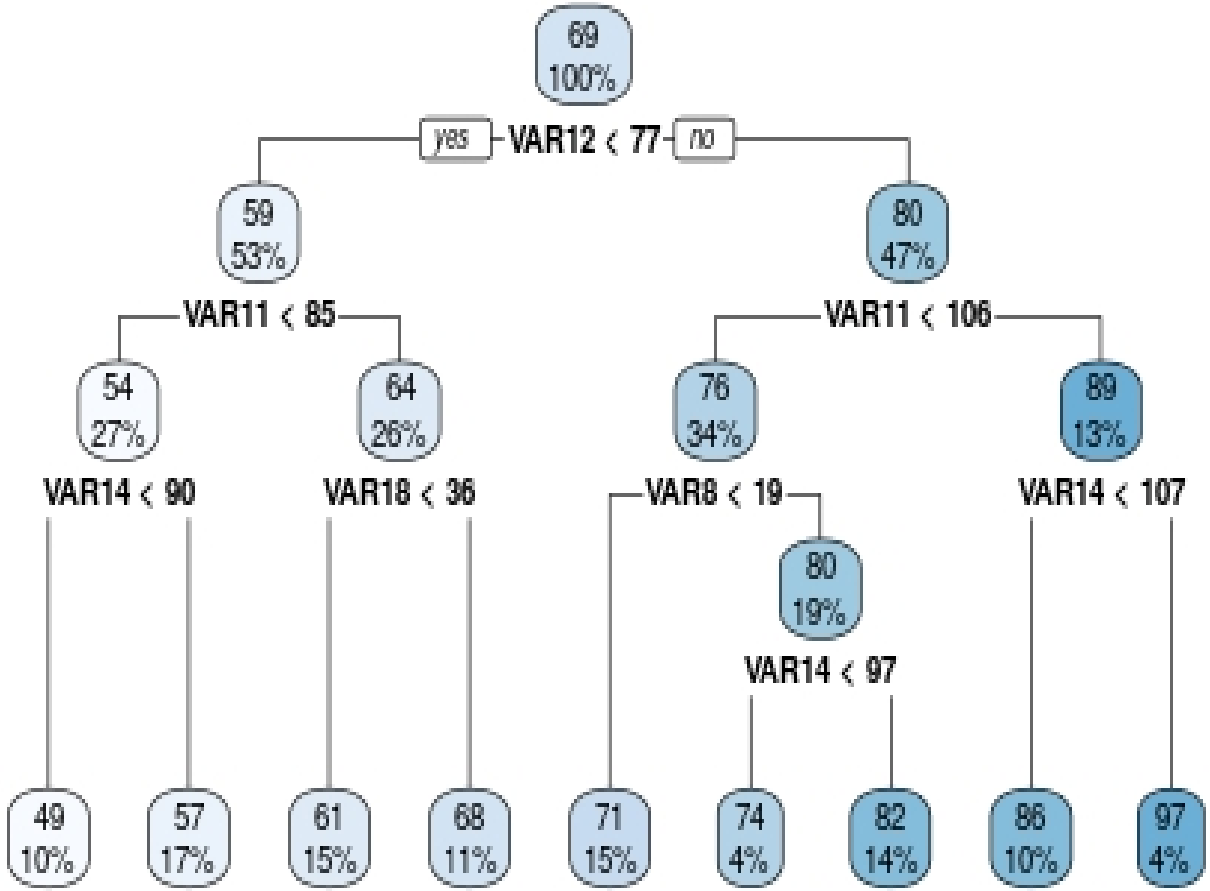


Figure: Body Dimension Data

Implementation using *rpart*

```
library(rsample)
library(rpart)
library(rpart.plot)
%library(Metrics) #rmse function

body_dimension_data_interest <- read.csv("~/Desktop/Workshop_stat_ml/
View(body_dimension_data_interest)

set.seed(123)
bodydim_split<-initial_split(body_dimension_data_interest, propo=.7)
bodydim_train<-training(bodydim_split)
bodydim_test<-testing(bodydim_split)

m1<-rpart(formula=weight~., data=bodydim_train, method="anova")
rpart.plot(m1) #to view the tree
plotcp(m1) #To check for the prunning

m2 <- rpart(formula=weight ~ .,data= bodydim_train, method= "anova",c
plotcp(m2)

pred <- predict(m1, newdata = bodydim_test)
obs<-bodydim_test$weight
rmse(pred, obs)
```

Ilustration using *rattle()*



Advantage/ Disadvantage

- Advantages

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity
  - Tree-method is a non parametric method (assumptions free)

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity
  - Tree-method is a non parametric method (assumptions free)
- Disadvantages

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity
  - Tree-method is a non parametric method (assumptions free)
- Disadvantages
  - High variance caused by the hierarchical nature of the process

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity
  - Tree-method is a non parametric method (assumptions free)
- Disadvantages
  - High variance caused by the hierarchical nature of the process
  - Lack of smoothness of the predictor surface (MARS alleviate)

Advantage/ Disadvantage

- Advantages
  - Trees are easy to interpret
  - Trees can handle multicolinearity
  - Tree-method is a non parametric method (assumptions free)
- Disadvantages
  - High variance caused by the hierarchical nature of the process
  - Lack of smoothness of the predictor surface (MARS alleviate)
  - Difficulty in modeling additive structure (MARS capture).

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.
- For MARS a similar process is used to find the best split with reference to the deviances from a spline function on either side of the split.

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.
- For MARS a similar process is used to find the best split with reference to the deviances from a spline function on either side of the split.
- The spline functions used by MARS are:

$$(x-t)_+ = \begin{cases} x-t & x > t \\ 0 & \text{o/w} \end{cases} \quad \text{and} \quad (t-x)_+ = \begin{cases} t-x & t > x \\ 0 & \text{o/w} \end{cases}$$

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.
- For MARS a similar process is used to find the best split with reference to the deviances from a spline function on either side of the split.
- The spline functions used by MARS are:
  $(x-t)_+ = \begin{cases} x-t & x > t \\ 0 & \text{o/w} \end{cases}$ and $(t-x)_+ = \begin{cases} t-x & t > x \\ 0 & \text{o/w} \end{cases}$
- Each function is a piecewise linear. By multiplying these splines together it is possible to produce quadratic or cubic curves.

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.
- For MARS a similar process is used to find the best split with reference to the deviances from a spline function on either side of the split.
- The spline functions used by MARS are:
  $(x-t)_+ = \begin{cases} x-t & x>t \\ 0 & \text{o/w} \end{cases}$ and $(t-x)_+ = \begin{cases} t-x & t>x \\ 0 & \text{o/w} \end{cases}$
- Each function is a piecewise linear. By multiplying these splines together it is possible to produce quadratic or cubic curves.
- The pair of functions $(X-t)_+$, $(t-X)_+$ is called reflected pair while t is called a knot.

MARS

- For the regression tree process, the data were partitioned in a way that produced the "best" split with reference to the deviances from the mean on either side of the split.
- For MARS a similar process is used to find the best split with reference to the deviances from a spline function on either side of the split.
- The spline functions used by MARS are:
  $(x-t)_+ = \begin{cases} x-t & x>t \\ 0 & \text{o/w} \end{cases}$ and $(t-x)_+ = \begin{cases} t-x & t>x \\ 0 & \text{o/w} \end{cases}$
- Each function is a piecewise linear. By multiplying these splines together it is possible to produce quadratic or cubic curves.
- The pair of functions $(X-t)_+$, $(t-X)_+$ is called reflected pair while t is called a knot.
- Recall: Regression tree uses as basis functions: $I(X_j > c)$ and $I(X_j \leq c)$

Illustrative Example

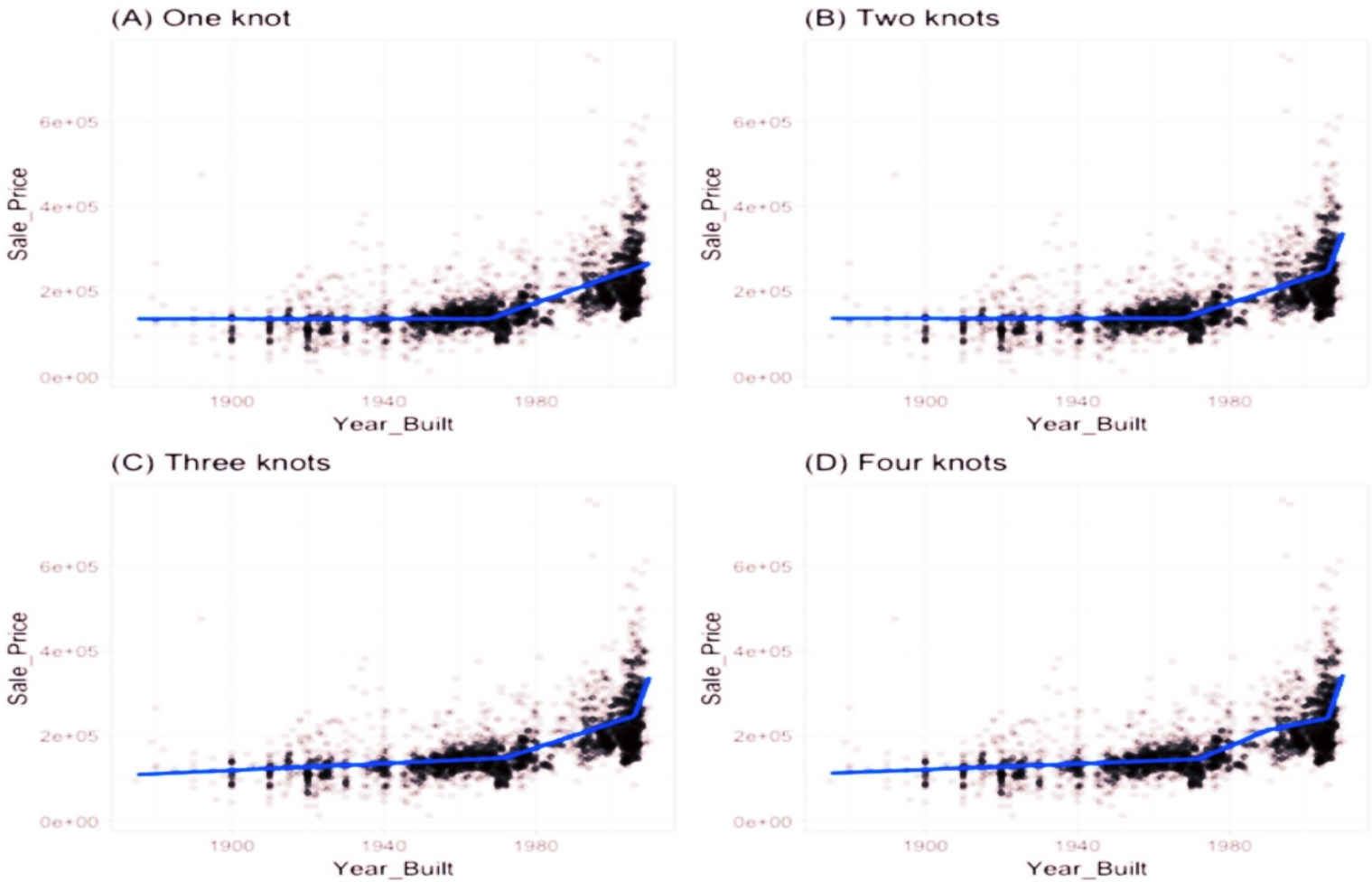


Figure 2: Examples of fitted regression splines of one (A), two (B), three (C), and four (D) knots.

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass
    - $f(X) = \beta_0 + \sum_{i=1}^{M} \beta_m h_m(X)$

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass
    - $f(X) = \beta_0 + \sum_{i=1}^{M} \beta_m h_m(X)$
    - Each $h_m(x)$ is either a function in $\mathcal{C}$ or product of two or more such functions

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass
    - $f(X) = \beta_0 + \sum_{i=1}^{M} \beta_m h_m(X)$
    - Each $h_m(x)$ is either a function in $\mathcal{C}$ or product of two or more such functions
    - In each step β_m's are estimated by minimizing the residual sum of squares

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass
    - $f(X) = \beta_0 + \sum_{i=1}^{M} \beta_m h_m(X)$
    - Each $h_m(x)$ is either a function in $\mathcal{C}$ or product of two or more such functions
    - In each step β_m's are estimated by minimizing the residual sum of squares
  - The backward pass

Multivariate Adaptive Regression Splines

- Data: $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})'$
- Consider basis functions of form
  $\mathcal{C} = \{(X_j - x_{ij})_+, (x_{ij} - X_j)_+\}$ $\begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p \end{matrix}$
- Model building strategy
  - The forward pass
    - $f(X) = \beta_0 + \sum_{i=1}^{M} \beta_m h_m(X)$
    - Each $h_m(x)$ is either a function in $\mathcal{C}$ or product of two or more such functions
    - In each step β_m's are estimated by minimizing the residual sum of squares
  - The backward pass
    - It prunes the model into its most effective part

The forward pass

- Step 1: Start with $h_0(X) = 1$; $\hat{f}^{(1)} = \hat{\beta}_0^{(1)}, \mathcal{M}^{(1)} = \{h_0(X)\}$

The forward pass

- Step 1: Start with $h_0(X) = 1$; $\hat{f}^{(1)} = \hat{\beta}_0^{(1)}, \mathcal{M}^{(1)} = \{h_0(X)\}$
- Step 2: Add to the model a function of the form $b_1(X_j - t)_+ + b_2(t - X_j)_+$, with $t \in \{x_{1j}, \ldots, x_{Nj}\}$ that produces the largest decrease in training error. Say this is achieved by $j = J$, and $t = x_{kJ}$.
  Model: $\hat{f}^{(2)} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)}(X_J - x_{kJ})_+ + \hat{\beta}_2^{(2)}(x_{kJ} - X_J)_+$,
  $\mathcal{M}^{(2)} = \{h_0(X), h_1(X), h_2(X)\}, h_1(X) = (X_J - x_{kJ})_+$ etc.

The forward pass

- Step 1: Start with $h_0(X) = 1$; $\hat{f}^{(1)} = \hat{\beta}_0^{(1)}, \mathcal{M}^{(1)} = \{h_0(X)\}$
- Step 2: Add to the model a function of the form $b_1(X_j - t)_+ + b_2(t - X_j)_+$, with $t \in \{x_{1j}, \ldots, x_{Nj}\}$ that produces the largest decrease in training error. Say this is achieved by $j = J$, and $t = x_{kJ}$.
  Model: $\hat{f}^{(2)} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)}(X_J - x_{kJ})_+ + \hat{\beta}_2^{(2)}(x_{kJ} - X_J)_+$,
  $\mathcal{M}^{(2)} = \{h_0(X), h_1(X), h_2(X)\}, h_1(X) = (X_J - x_{kJ})_+$ etc.
- ...

The forward pass

- Step 1: Start with $h_0(X) = 1$; $\hat{f}^{(1)} = \hat{\beta}_0^{(1)}, \mathcal{M}^{(1)} = \{h_0(X)\}$
- Step 2: Add to the model a function of the form $b_1(X_j - t)_+ + b_2(t - X_j)_+$, with $t \in \{x_{1j}, \ldots, x_{Nj}\}$ that produces the largest decrease in training error. Say this is achieved by $j = J$, and $t = x_{kJ}$.
  Model: $\hat{f}^{(2)} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)}(X_J - x_{kJ})_+ + \hat{\beta}_2^{(2)}(x_{kJ} - X_J)_+$,
  $\mathcal{M}^{(2)} = \{h_0(X), h_1(X), h_2(X)\}, h_1(X) = (X_J - x_{kJ})_+$ etc.
- ...
- Step m + 1: Add to the model a function of the form $b_{2m-1}h_l(X)(X_j - t)_+ + b_{2m}h_l(X)(t - X_j)_+$ with $h_l(X) \in \mathcal{M}^{(m)}$ that produces the largest decrease in training error. Say this is achieved by $j = J', t = x_{k'J'}$ and $l = L$.
  $\mathcal{M}^{(m+1)} = \mathcal{M}^{(m)} \cup \{h_{2m-1}(X), h_{2m}(X)\}$
  $h_{2m-1} = h_L(X)(X_{J'} - x_{k'J'})_+$ and $h_{2m} = h_L(X)(x_{k'J'} - X_{J'})_+$

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms
- The model typically overfits the data including large number of terms

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms
- The model typically overfits the data including large number of terms
- Backward pass prunes the model

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms
- The model typically overfits the data including large number of terms
- Backward pass prunes the model
- It deletes the least effective terms one by one

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms
- The model typically overfits the data including large number of terms
- Backward pass prunes the model
- It deletes the least effective terms one by one
- Generalized cross validation is used to choose the size of the model and the most effective subset of terms

The backward pass

- The forward pass algorithm stops when the model set contains some preset number of terms.
- Stopping rule: One can fix the **degree** of interaction terms
- The model typically overfits the data including large number of terms
- Backward pass prunes the model
- It deletes the least effective terms one by one
- Generalized cross validation is used to choose the size of the model and the most effective subset of terms
- $GCV(\lambda) = \frac{\sum_{i=1}^{n}(y_i - f_\lambda(x_i))^2}{(1-M(\lambda)/n)^2}$, where λ is size of the model and $M(\lambda)$ is the effective number of parameters in the model.

Implementation of MARS in R

```
#implementation of MARS
library(earth)
library(caret)

mars1 <- earth(weight ~ ., data = bodydim_train)
print(mars1) #for model summary
summary(mars1)
plot(mars1, which = 1)

mars2 <- earth(weight ~., data = bodydim_train, degree = 2)
summary(mars2)
plot(mars2,which=1,legend.pos=0)

#performance of the model in test dataset
summary(mars1,newdata=bodydim_test)
yhat=predict(mars1,newdata=bodydim_test)
yobs=bodydim_test$weight
rmse(yhat, yobs)
```

Advantage and Disadvantage of MARS

- Advantages:
  - Accurate if the local linear relationships are correct.
  - Quick computation.
  - Can work well even with large and small data sets.
  - Provides automated feature selection.
  - The non-linear relationship between the features and response are fairly intuitive.
  - Can be used for both regression and classification problems.
  - Does not require feature standardization.
- Disadvantages:
  - Not accurate if the local linear relationships are incorrect.
  - Typically not as accurate as more advanced non-linear algorithms (random forests, gradient boosting machines).
  - The earth package does not incorporate more advanced spline features (i.e. Piecewise cubic models).
  - Missing values must be pre-processed.

Thank You