

Classification and Regression trees

(Recursive partitioning)

N. Torelli, G. Di Credico, V. Gioia
2023

University of Trieste

Regression Trees

Classification Trees

MARS: Multivariate Adaptive Regression Splines

Regression Trees

Regression trees are a machine learning technique used for predicting a continuous response variable by recursively partitioning the predictor space. They are part of the broader family of decision trees, but specifically focus on regression tasks rather than classification.

Step functions as approximators

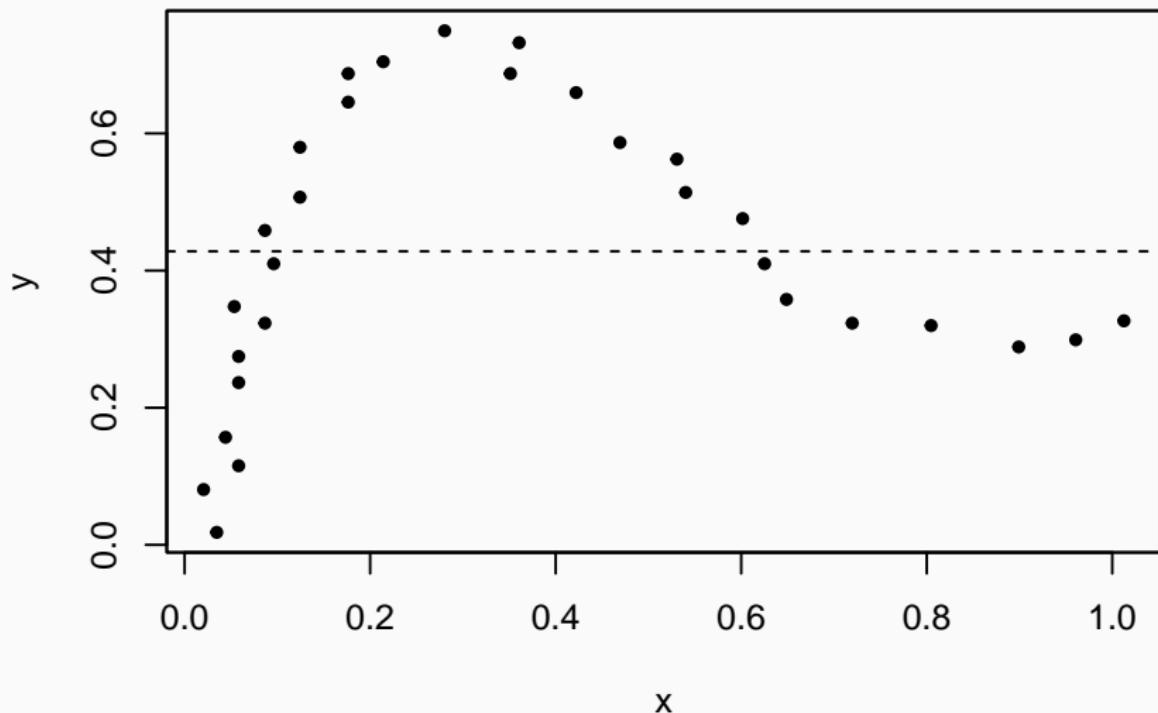
- A simple, yet effective, way to approximate a generic function $f(x)$ is to use a step function, that is, a piecewise constant function
- In such a case, there are various choices to be made:
 - where are the subdivision points to be placed?
 - which value of y must be assigned to each interval?
 - how many subdivisions of the x axis must be considered?
- The idea is to generalize the use of step functions to approximate (or predict) a response Y as function of some covariates.
- Note that Y could be of different nature: numeric, factor, counts

Step functions as a spline

- A step function actually is a spline of degree 0. Assume we want to fit such a function to a simple set of data.
- Subdivision points are now the knots and their position should be chosen to reflect changes of the function $f(x)$ (for instance more knots where the function is steeper)
- In a given interval the value of the constant can be chosen to be an average of the level of the function itself
- The choice of the number of subdivisions is critical: any increase in the number of steps increases the quality of the approximation, and therefore we are led to think of infinite subdivisions.
- However, this is counter to the requirement to use a approximate representation using few parameters and therefore to adopt a finite number of subdivisions.

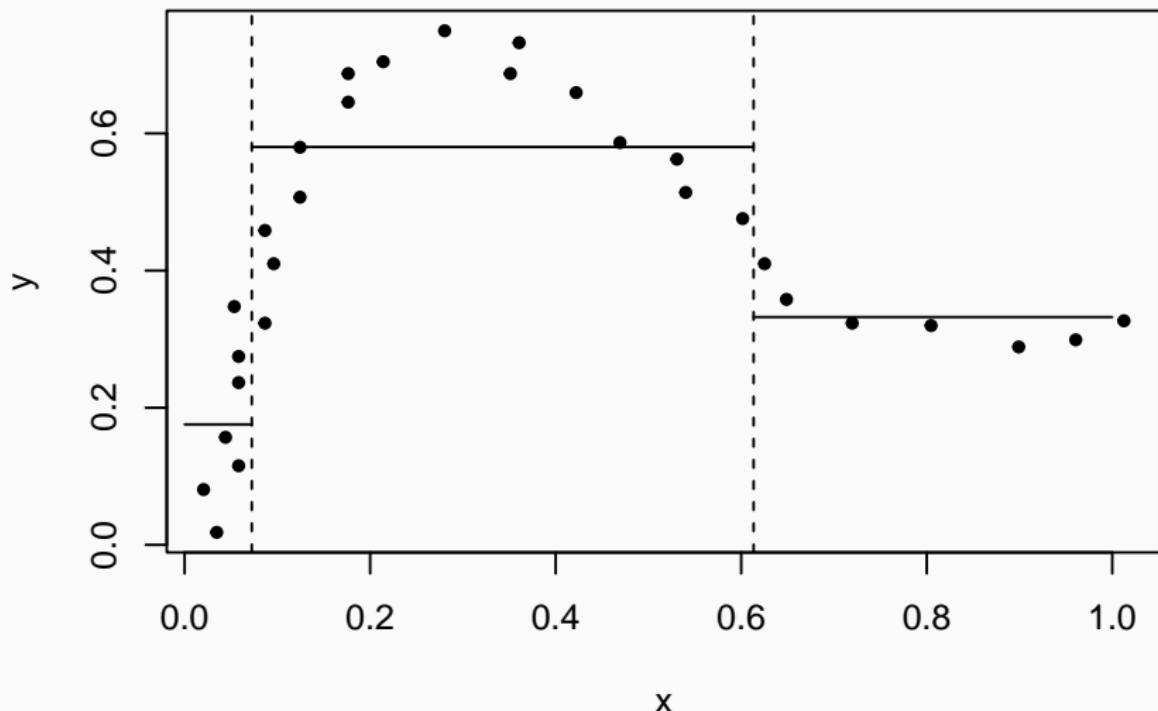
An introductory example

- If y is quantitative a global approximation of y could be its mean. Or we can use a (regression) function $g(\cdot)$



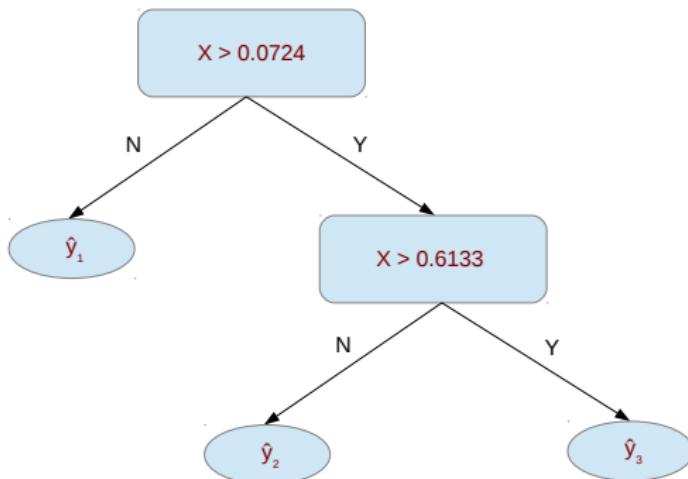
An introductory example

- Now consider a subdivision on X and approximate y with its local mean \hat{y}_i in the i -th interval and g is a piecewise constant function



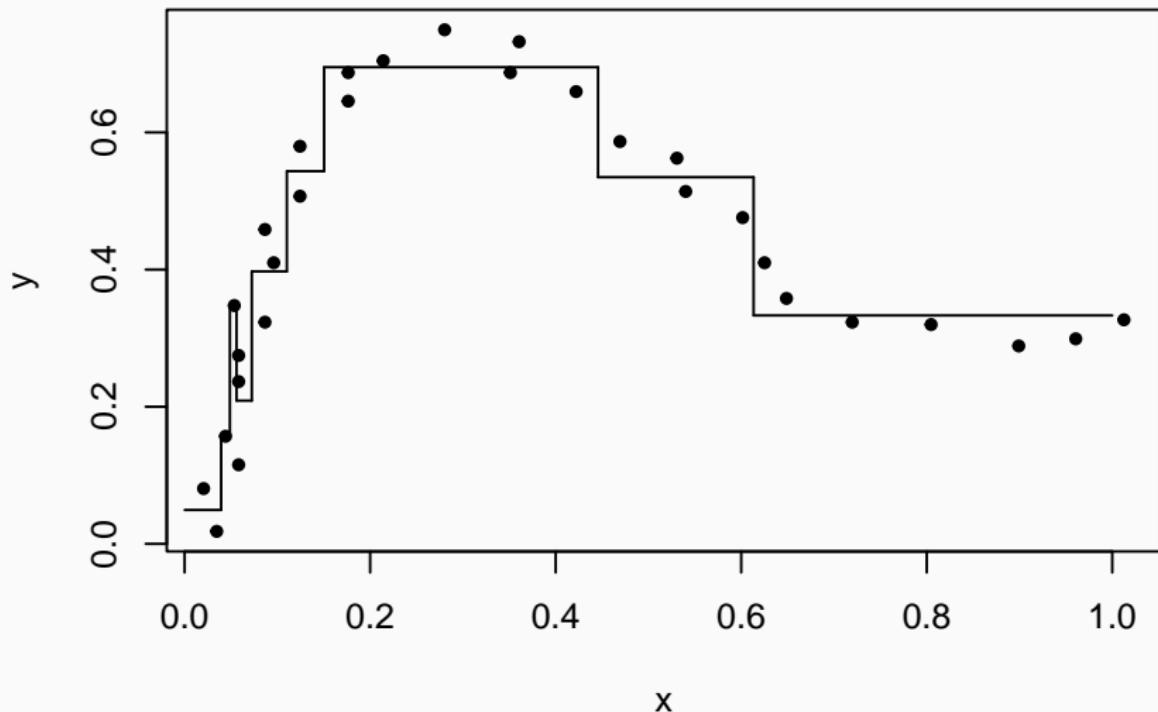
The tree

Note that the value \hat{y}_i of the function g can be also described by the following tree



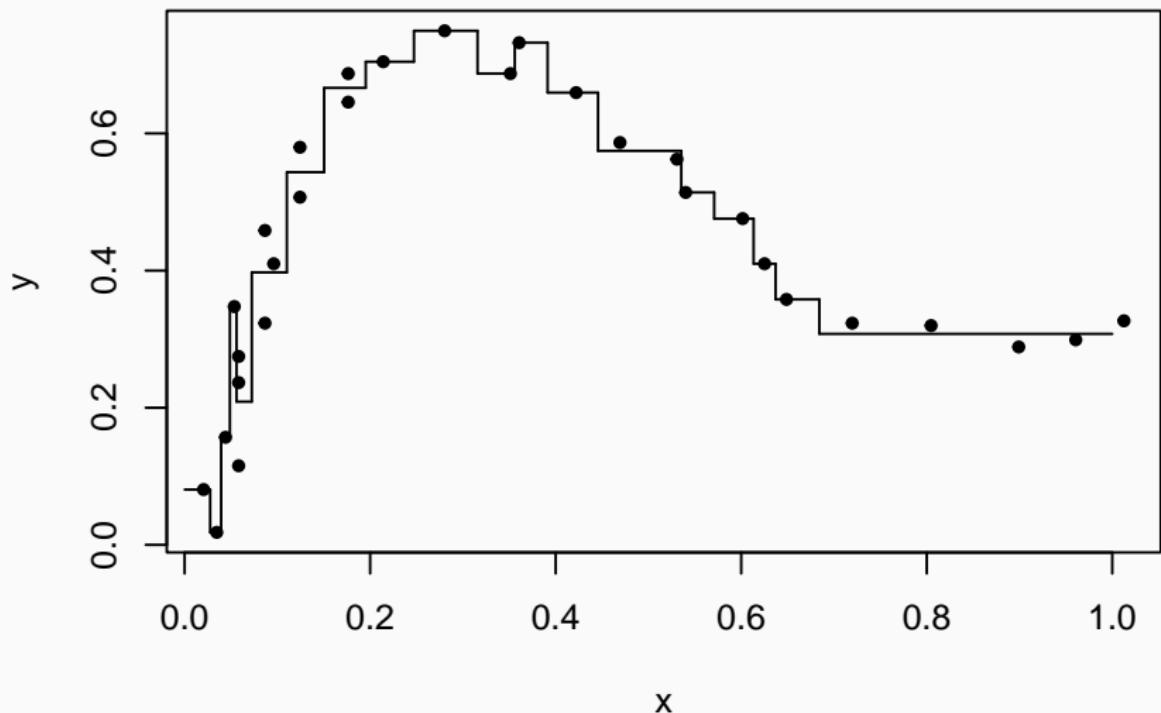
An introductory example

- As the number of intervals increase, we could achieve a very accurate description of the data



An introductory example

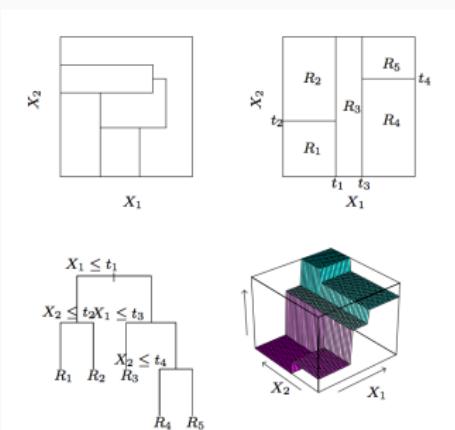
- As the number of intervals increase, we could achieve a very accurate description of the data (leading to overfitting)



Tree approximation

- Let's now consider a regression problem with continuous response Y and two covariates X_1 and X_2 . We want to estimate the generic regression curve $E(Y) = f(x_1, x_2)$.
- The idea is again to partition the space spanned by the covariates and to model Y with a different constant in each element of the partition
- we restrict attention to recursive binary partitions.
 - First split the space into two regions, and model the response by the mean of Y in each region.
 - variable and split-point are chosen in order to achieve the best fit.
 - one or both of these regions are split into two more regions,
 - the process is continued, until some stopping rule is applied.

A simple example of tree partitioning for two covariates



In the top right panel first split at $X_1 = t_1$. Then the region $X_1 \leq t_1$ is split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$. Finally, the region $X_1 > t_3$ is split at $X_2 = t_4$. The result of such a recursive buinary splitting is a partition into the five regions R_1, R_2, \dots, R_5 shown in the figure.

- The corresponding regression model predicts Y with a constant c_m in region R_m , that is, $\hat{f}(X_1, X_2) = \sum_{m=1}^5 c_m I\{(X_1, X_2) \in R_m\}$
- The sets R_m are rectangles, in the 2-dimensional space, with their edges parallel to the coordinate axes) and c_1, \dots, c_5 are constants. Note that the top left panel represents a partition that cannot be obtained by recursive binary splitting

A regression tree

- More generally:
 - we want estimate a regression curve $f(x_1, x_2, \dots, x_p)$ underlying the data by $\hat{f}(x_1, x_2, \dots, x_p) = \sum_{m=1}^M c_m I\{(x_1, x_2, \dots, x_p) \in R_m\}$ where $I(x_1, x_2, \dots, x_p \in R_m)$ is the indicator function of the set R_m (R_m are rectangles, in the p -dimensional sense, with their edges parallel to the coordinate axes) and c_1, \dots, c_M are constants.
 - Given an objective function such as the Deviance

$$D = \sum_{i=1}^n (y_i - \hat{f}(x_{1i}, x_{2i}, \dots, x_{pi}))^2$$

- the goal is to define a partition of the space of the covariates that minimizes D

Building the Regression tree

- this minimization, even if we fix the number of the elements of the partition, involves very complex computation
- a sub-optimal approach is considered using a step-by-step optimization: we construct a sequence of gradually more refined approximations and to each of these we minimize the deviance relative to the passage from the current approximation to the previous one
- It is not ensured that we get the global maximum. This procedure is called greedy-algorithm
- This operation is represented by a series of binary splits
- Each internal node represents a value query on one of the variables – e.g. “Is $x_3 > 0.4?$ ”. If the answer is ‘Yes’, go right, else go left.
- The terminal nodes are the decision nodes. Typically each terminal node is assigned a value, c_h , given by the arithmetic mean of the observed y_i having component x_{ji} falling in this node.

Growing the tree

- Trees are grown using a random subset of the available data (*the training data*), by recursive splitting
- A terminal node g is split into the left and right daughters (g_L and g_R) that increase the split criterion

$$D_g - D_{g_L} - D_{g_R}$$



the most, where D is the deviance associated to a given node.

- To avoid the overfitting, a large tree T_0 is grown and then pruned backward
- Indeed a tree with n leaves is equivalent to a polynomial regression of degree $n - 1$
- detection of the variable X_j that achieve the best split at each node and which is the split point can be done very quickly and hence by scanning through all of the inputs
- Deviance can be adapted for dealing with a response that is a count or a duration

Pruning the tree

- Pruning criterion: cost of a subtree $T \in T_0$, is defined by

$$C_\alpha(J) = \sum_{j=1}^J D_j + \alpha_j$$

T_0 IS GROWN TO ITS MAXIMUM DEPTH, WHERE EACH TERMINAL NODE REPRESENTS A REGION WITH VERY FEW OBS.
AVOID OVERFIT FOR SRC.

D_j : REGIONAL SUM OF SQUARES IN TERMINAL NODE

- Here the sum is over the terminal nodes of T , J is the number of terminal nodes in T and α is a cost-complexity parameter \rightsquigarrow (CONTROL THE TRADE-OFF BETWEEN FIT AND COMPLEXITY)
- The choice of an optimal size is evaluated by cross-validation, or on a validation set.
- For each α the best subtree T_α is found via weakest link pruning
- Larger α gives smaller trees AVOIDING OVERFIT WHEN THE DATA IS NOISY OR SPARSE
- A best value $\hat{\alpha}$ is estimated via cross-validation (or on a validation set)
- Final chosen tree is $T_{\hat{\alpha}}$
- New observations are classified by passing their x down to a terminal node of the tree, and then using the relative c_h .

An example

The variable FACE refer to the amount of life insurance bought by the head of a household. We want to predict it by using “INCOME”, number of household members, AGE, Education, etc. For illustration, a tree with maximum depth=2 is considered.

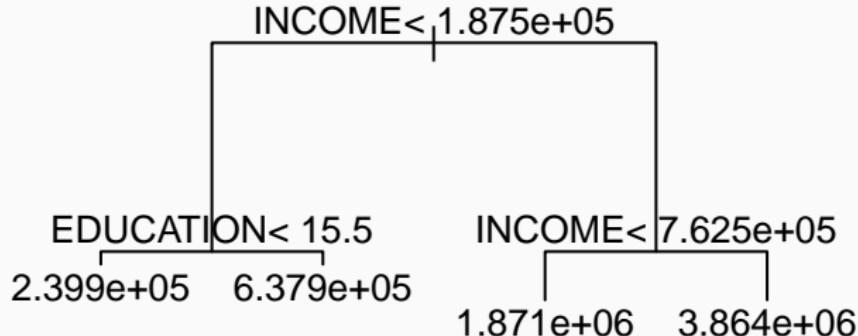
Package rpart is used.

```
TL <- read.csv("TL.csv", header=TRUE, sep=",", row.names=1)
library(rpart)
attach(TL)

m2 <- rpart(FACE~INCOME+MARSTAT+NUMHHH+EDUCATION+AGE,
             control=rpart.control(maxdepth=2))
m2

## n= 275
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 275 7.681561e+14  747581.5
##   2) INCOME< 187500 227 2.629158e+14  413511.5
##     4) EDUCATION< 15.5 128 1.075360e+14  239930.5 *
##     5) EDUCATION>=15.5 99 1.465367e+14  637939.4 *
##   3) INCOME>=187500 48 3.600986e+14  2327454.0
##     6) INCOME< 762500 37 1.905974e+14  1870751.0 *
##     7) INCOME>=762500 11 1.358255e+14  3863636.0 *
```

The tree



Classification Trees

Classification Trees

- If the target (response) variable is a categorical variable taking values $1, 2, \dots, K$, the only changes needed in the tree algorithm pertain to the criteria for splitting nodes and possibly pruning the tree.
- In these cases the tree will be used for predicting the categorical response and this is labeled as a classification problem. And the tree is then a **Classification tree**.
- Also in this case a tree is a hierarchical structure formed by:
 - root: the predictor space
 - nodes:
 1. internal: test an explanatory variable (and splits the predictor space)
 2. terminal (leaf): assign a label class
 - branches: corresponds to values of the explanatory variables
- A tree is constructed by repeated splits of the predictor space (root) into subregions (nodes). Each terminal region is associated with a prediction and their union form a partition of the predictor space.

Growing a classification tree

The following elements are needed

- A set of splits
- A goodness of split criterion
- A stop-splitting rule
- A rule for assigning every terminal node to a class
- Each split depends on the value of a single predictor x_j and depends on the nature of x_j :
 - qualitative, with values in $\mathcal{L} = \{l_1, \dots, l_K\}$: a split is any question as “ $x_j \in S_{\mathcal{L}}$?” with $S_{\mathcal{L}}$ a subset of \mathcal{L} ;
 - quantitative, with range (a, b) : a split is any question as “ $x_j \leq s$?” with $a \leq s < b$
- Examples
 - “Is the age of the subject not greater than 60?”
 - “Is the weather cloudy or rainy?”
- At each step of the tree growing procedure, the best split is identified for each predictor and, among these, the best of the best is selected.

The goodness of split criterion

- The objective of classification tree construction is to finally obtain nodes that are as pure as possible, i.e., the split should send towards each branch observations of the same class
- It makes sense to consider good a split when it leads to a high reduction of impurity of the node (a high increase of the prediction/classification accuracy).
- Consider a node t for a two class classification problem, the two classes of y have frequency $p(t)$ and $1 - p(t)$. An impurity measure of a node t is a function of the proportion of units into the two classes. Let us consider the misclassification error defined as

$$i(t) = 1 - \max(p(t), (1 - p(t)))$$

as an impurity measure

- If the node is equipped with a split sending a proportion of p_L and p_R to the left and, respectively right, the gained reduction of impurity is:

$$\Delta i(t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

- The best split is the split which maximizes the reduction of impurity

Impurity measures

More generally, for a multiclass problem, for a given node m that defines a region R_M with N_M observations, \hat{p}_{mk} is the observed proportion of cases in class k . The observation at the node will be classified in class $k(m)$ that is the class for which \hat{p}_{mk} is larger. The following impurity measures can be defined:

- Misclassification error: *NUMBER OF MISSCLASSIFIED OBSERVATION IN A REGION*

$$\frac{1}{N_M} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$$

- Gini index (heterogeneity index): *MEASURES IMPURITY WITHIN A REGION*

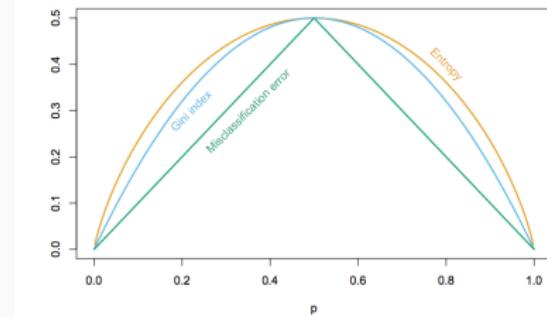
$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad \text{MINIMIZE THIS WEIGHT}$$

- Entropy: *MEASURES THE "DISORDER" K OF A RESPONSE VARIABLE IN A REGION*

$$H = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk} \quad \text{MAXIMIZE THE ENTROPY}$$

Measures of impurity in two-class problems

- for $K = 2$, with p the observed proportion in the second class, these three measures are respectively:
 - $1 - \max(p, 1 - p)$
 - $2p(1 - p) = 2(p - p^2)$
 - $-p \log p - (1 - p) \log(1 - p)$



Avoiding overfitting

- If the overall accuracy is too low we may always make the tree growing further
- The flexibility of the trees would in principle allow for building a perfect classification rule
- A tree that perfectly fits the sample data probably overfits the data: useless for predicting new data, not used for training the tree!
- A useful practice is to evaluate the accuracy of the estimated tree on a test set (out-of-sample).
- Often for Regression and Classification trees the available data are randomly subdivided into three sets:
 - the training set (to grow the tree)
 - the validation set (to prune it)
 - the test set (to evaluate it)
- Evaluation of the quality of the three can be achieved with usual tools for evaluating the prediction (classification) quality: Mean squared prediction errors, confusion matrices, ROC curves (see the R package ‘caret’)

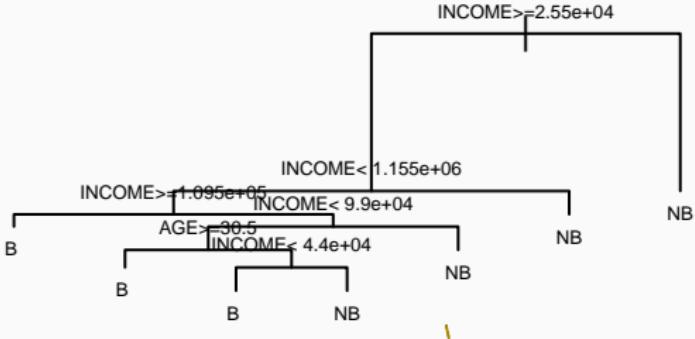
An example of two class tree

We want to predict now if a life insurance policy is bought using the same covariates

```
TL <- read.csv("TLbin.csv", header=TRUE, sep=",", row.names=1);
attach(TL); set.seed(4321); ind.train <- sample(1:500,300) ;
TL.train <- TL[ind.train,]; TL.test <- TL[-ind.train,]
tree <- rpart(FACEPOS~., data=TL.train); tree

## n= 300
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
##  1) root 300 136 B (0.5466667 0.4533333)
##    2) INCOME>=25500 235  90 B (0.6170213 0.3829787)
##      4) INCOME< 1155000 227  84 B (0.6299559 0.3700441)
##        8) INCOME>=109500 72  19 B (0.7361111 0.2638889) *
##        9) INCOME< 109500 155  65 B (0.5806452 0.4193548)
##       18) INCOME< 99000 145  58 B (0.6000000 0.4000000)
##         36) AGE>=30.5 122  45 B (0.6311475 0.3688525) *
##         37) AGE< 30.5 23  10 NB (0.4347826 0.5652174)
##           74) INCOME< 44000 14   5 B (0.6428571 0.3571429) *
##           75) INCOME>=44000 9   1 NB (0.1111111 0.8888889) *
##           19) INCOME>=99000 10   3 NB (0.3000000 0.7000000) *
##           5) INCOME>=1155000 8   2 NB (0.2500000 0.7500000) *
##           3) INCOME< 25500 65  19 NB (0.2923077 0.7076923) *
```

The tree



```
pred.test <- predict(tree, newdata=TL.test, type="class")
t <- table(TL.test$FACEPOS, pred.test)
t

##      pred.test
##      B NB
##      B 78 33
##      NB 49 40
sum(diag(t))/sum(t)

## [1] 0.59
```

Dealing with missing data

- It is quite common to have observations with missing values for one or more input features. The usual approach in statistics is to impute (fill-in) the missing values in some way.
- However, the first issue in dealing with missing data is whether the missing data introduce a sample selection that can bias results of analyses.
- It is important consider if missing data arise by a
 - Missing Completely at Random (MCAR) mechanism (*no bias*)
 - Missing at Random (MAR) mechanism (*possible bias if the dependence on missingness on some observed covariates are not recognized*)
 - Missing Not at Random (MNAR) mechanism (*huge problems, likely to have non negligible bias*)
- For the first, and possibly, the second case, in regression trees two approaches can be used when predictors have missing values:
 - if it is categorical, add a specific category for missing values
 - if it is continuous, use surrogate predictors to be used when observation is missing on the primary predictor.

Regression and classification trees: Advantages

- Logical simplicity and ease of 'communication' (particularly those with a non-quantitative background)
- The step function has a simple, compact mathematical formulation in terms of information to be stored
- Speed of computation and can take advantage of parallel calculation
- Can handle huge datasets
- Can handle mixed predictors: quantitative and factors
- Easy ignore redundant variables and automatically detects interactions among variables
- Handle missing data elegantly
- Small trees are easy to interpret

Regression and classification trees: Disadvantages

- Instability of results: very sensitive to the insertion/changes in the sample
- Difficulty in upgrading: if more data arrive, they cannot be added to the already constructed tree; it is necessary to start again from the beginning.
- Difficulty of approximating some mathematically simple functions, particularly if they are steep,
- Statistical inference: formal procedures of statistical inference such as hypothesis testing, confidence intervals, and others are not available.
- (over?) emphasizes interactions
- large trees are hard to interpret
- prediction surface is not smooth

MARS: Multivariate Adaptive Regression Splines

MARS: Multivariate Adaptive Regression Splines

- MARS is an adaptive procedure for regression, and is well suited for high dimensional problems (i.e., a large number of inputs).
- It can be viewed as a generalization of stepwise linear regression or a modification of the CART. This latter approach for regression tree leads to smoother prediction surfaces
- A hybrid of MARS called PolyMARS specifically designed to handle classification problems has been also proposed
- MARS is a semi-parametric method that like CART uses a greedy algorithm and recursively adapt a curve to the regression surface
- At each step it is chosen a couple of basis functions recursively selecting the variable X that is most appropriate and the optimal position of the knot.

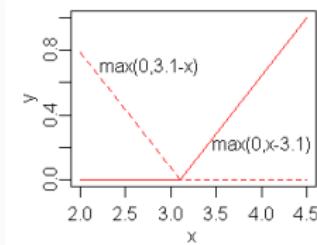
- MARS builds models of the form

$$\hat{f}(x) = \sum_{i=1}^k c_i B_i(x)$$

- The model is a weighted sum of basis functions $B_i(x)$. Each c_i is a constant coefficient.
- Each basis function $B_i(x)$ takes one of the following three forms:
 - a constant
 - a hinge function. A hinge function has the form $\max(0, x - const)$ or $\max(0, const - x)$.
MARS automatically selects variables and values of those variables for knots of the hinge functions.
 - a product of two or more hinge functions. These basis functions can model interaction between two or more variables.

MARS

This is an example of a couple of Hinge functions

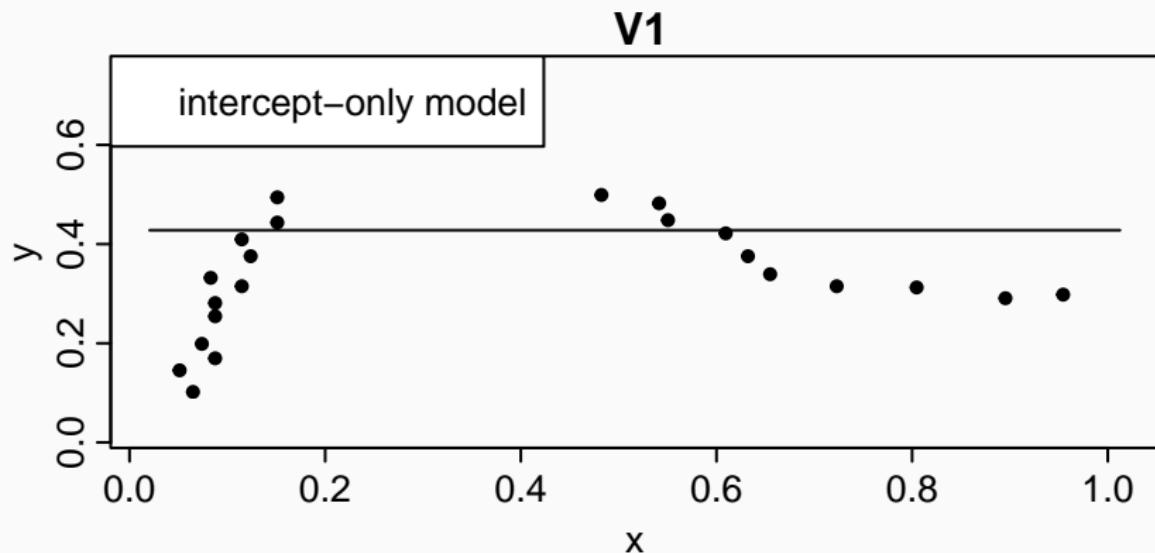


- Although they might seem quite different, the MARS and CART strategies actually have strong similarities.
- Suppose we take the CART procedure and make the following changes:
 - Replace step functions by the piecewise linear basis functions $I(x - t > 0)$ and $I(x - t \leq 0)$.
 - When a model term is involved in a multiplication by a candidate term, it gets replaced by the interaction, and hence is not available for further interactions.
 - With these changes, the MARS forward procedure is the same as the CART tree-growing algorithm.

An example

```
mod1=earth(V2~V1,data=x,nk=1)
plotmo(mod1,xlab="x",ylab="y")
points(x,pch=20)
```

V2 `earth(V2~V1, data=x, nk=1)`



An example

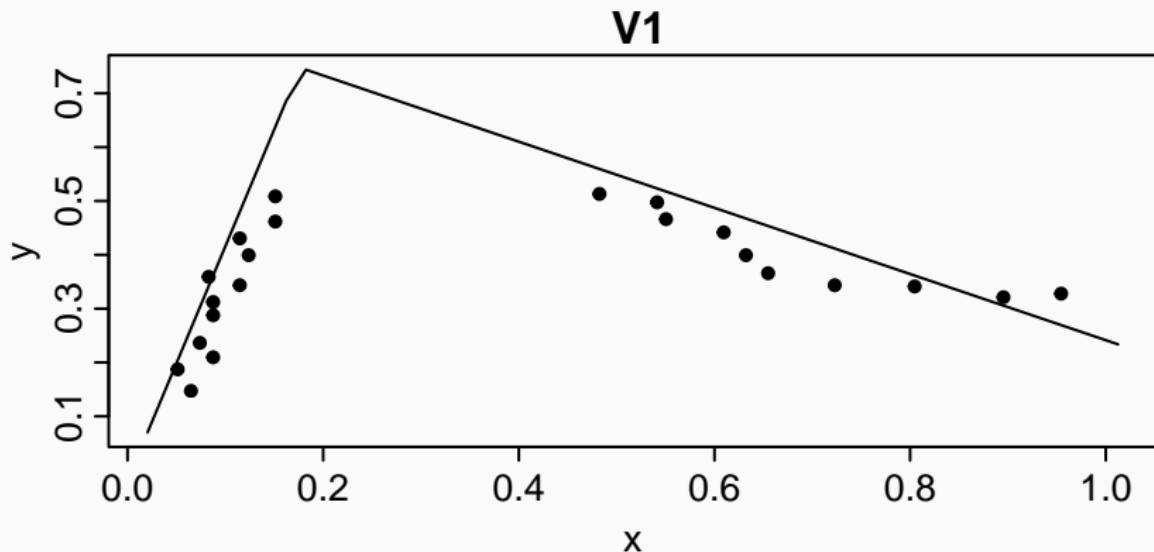
```
summary(mod1)

## Call: earth(formula=V2~V1, data=x, nk=1)
##
##           coefficients
## (Intercept)  0.4279076
##
## Selected 1 of 1 terms, and 0 of 1 predictors
## Termination condition: Reached nk 1
## Importance: V1-unused
## Number of terms at each degree of interaction: 1 (intercept only model)
## GCV 0.04290529    RSS 1.202778    GRSq 0    RSq 0
```

An example

```
mod2=earth(V2~V1,data=x,nk=4)
plotmo(mod2,xlab="x",ylab="y")
points(x,pch=20)
```

V2 earth(V2~V1, data=x, nk=4)



An example

```
summary(mod2)

## Call: earth(formula=V2~V1, data=x, nk=4)
##
##           coefficients
## (Intercept)      0.7476095
## h(0.176378-V1) -4.3458394
## h(V1-0.176378) -0.6146156
##
## Selected 3 of 3 terms, and 1 of 1 predictors
## Termination condition: Reached nk 4
## Importance: V1
## Number of terms at each degree of interaction: 1 2 (additive model)
## GCV 0.00632364    RSS 0.1317425    GRSq 0.852614    RSq 0.8904682
```