# Classification and Regression trees

(Recursive partitioning)

R. Bellio & N. Torelli

Spring 2018

University of Udine & University of Trieste

## Regression Trees

## Non-parametric regression mpdels

- **Non-parametric** (or semi-parametric) regression modelling keeps the usual specification:

$$y = g(x_1, \ldots, x_p, \epsilon)$$

  but relaxes the assumption of linearity, and replaces it with a much weaker assumption of a smooth $g$
- Pro's and con's
  - $\mapsto$ greater flexibility and potentially more accurate estimate of $g$
  - $\mapsto$ greater computation and often more difficult-to-interpret results: typically used for prediction, not interpretation
- Some examples of nonparametric regression models are:
  - $\mapsto$ Local Polynomial Regression
  - $\mapsto$ Kernel regression
  - $\mapsto$ Smoothing splines
  - $\mapsto$ (Generalized) Additive models
  - $\mapsto$ Decision (regression) trees

## Step functions as approximators

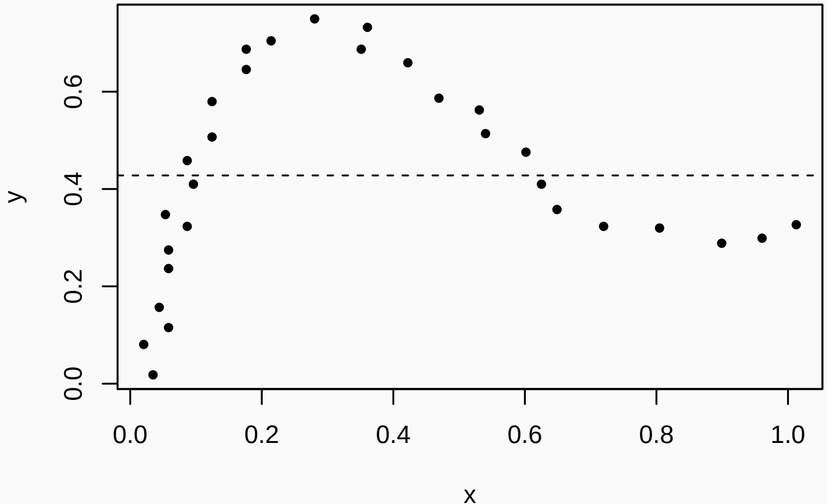
- A simple, yet effective, way to approximate a generic function $f(x)$ is to use a step function, that is, a piecewise constant function
- In such a case, there are various choices to be made:
  - where are the subdivision points to be placed?
  - which value of y must be assigned to each interval?
  - 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, count, ...

## 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 isntance more knots where the function is steeper)
- In a given interval the value of the constant can be chosen to be a 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.

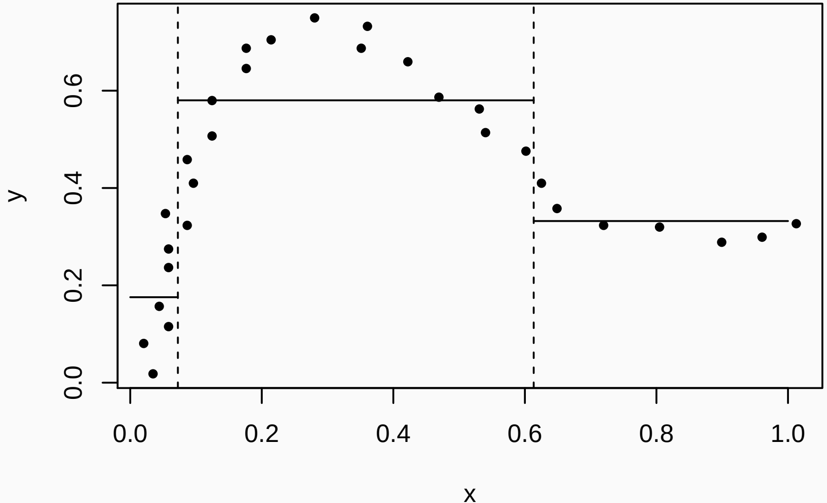
## An introductory examples

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



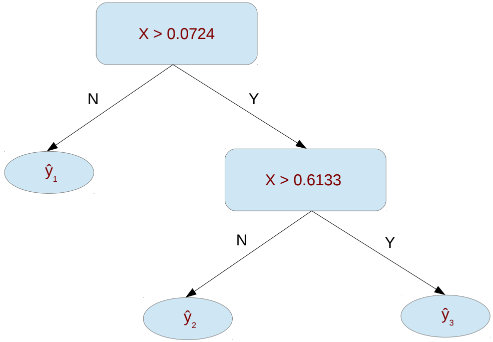
## An introductory examples

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



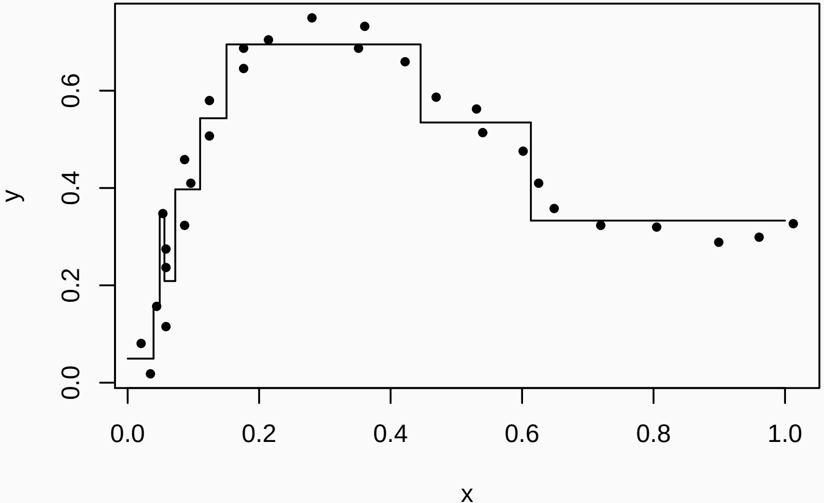
## The tree

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



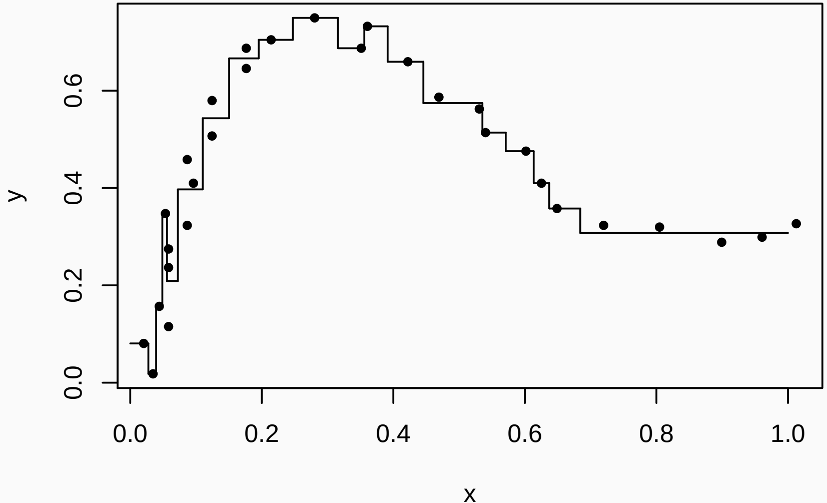
## An introductory examples

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

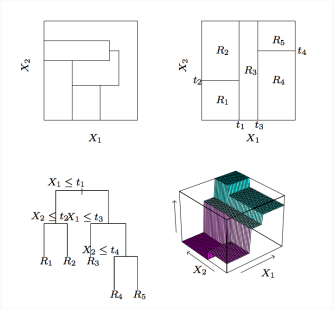


## An introductory examples

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



## A simple example of tree partitioning for two covariates



In the top right panel first split at $X_1 = t_1$. Then the region $X_1 \le t_1$ 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. 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 $f(x_1, x_2, \ldots, x_p)$ underlying the data by $\hat{f}(x_1, x_2, \ldots, x_p) = \sum_{m=1}^{M} c_m I\{(x_1, x_2, \ldots, x_p) \in R_m\}$ where $I(x_1, x_2, \ldots, x_p \in R_m)$ 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 $c_1, \ldots, c_M$ are constants.
  - Given an objective function such as the Deviance

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

  - 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 elemnts of the partition, involves very complex computation
- operatively a sub-optimal approach is considered, of 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

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

- Here the sum is over the terminal nodes of $T$, $J$ is the number of terminal nodes in $T$ and $\alpha$ is a cost-complexity parameter
- The choice of an optimal size is evaluated by cross-validation, or on a validation set.
- For each $\alpha$ the best subtree $T_\alpha$ is found via weakest link pruning
- Larger $\alpha$ gives smaller trees
- 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

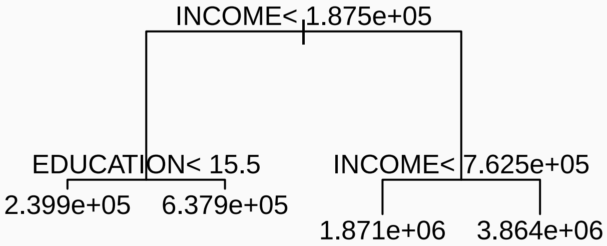
The variable FACE refer to the amount of life insurance bought by the head of a household. We want to predict it by using “INCOME”, number of household members, AGE, Education, etc. For illustration, a tree with maximum depth=2 is considered. Package rpart is used.

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

```
m2 <- rpart(FACE~INCOME+MARSTAT+NUMHH+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


INCOME< 1.875e+05
EDUCATION< 15.5
INCOME< 7.625e+05
2.399e+05
6.379e+05
1.871e+06
3.864e+06

## Regression 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 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

## Dealing with missing data

- It is quite common to have observations with missing values for one or more input features. The usual approach 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.
- It is important to make the distinction between cases where data are
  - 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) *(huge problems, likely to have non negligible selection 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.

## Classification Trees

## Classification Trees

- If the target (response) variable is a categorical variable taking values $1, 2, \ldots, K$, 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 $\mathcal{L} = \{l_1, \ldots, l_K\}$: a split is any question as "is $x_j \in S_{\mathcal{L}}$ ?" with $S_{\mathcal{L}}$ a subset of $\mathcal{L}$;
  - quantitative, with range $(a, b)$: a split is any question as "is $x_j \leq s$?" 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 classificatio problem, the two calsses of $y$ have frequency $p(t)$ and $1 - p(t)$. A natural **impurity measure** of a node $t$ is, the so called Misclassification error:

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

- 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
- Other measures of impurity could be used (Gini or Entropy based)

## Impurity measures

More generally, 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:

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

- Gini index (heterogeneity index):

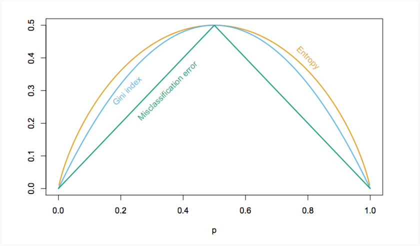
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- Entropy:

$$H = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

## 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)

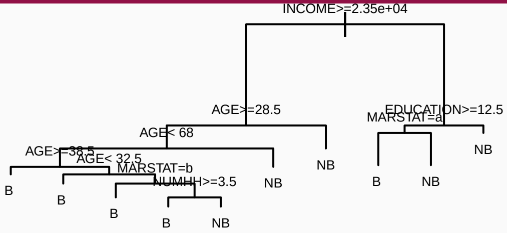
## An example of two class tree

We want to predict now if a life insurance policy is bougth 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 300 132 B (0.5600000 0.4400000)
##     2) INCOME>=23500 240  91 B (0.6208333 0.3791667)
##       4) AGE>=28.5 223  80 B (0.6412556 0.3587444)
##         8) AGE< 68 209  71 B (0.6602871 0.3397129)
##          16) AGE>=38.5 166  52 B (0.6867470 0.3132530) *
##          17) AGE< 38.5 43  19 B (0.5581395 0.4418605)
##            34) AGE< 32.5 15   4 B (0.7333333 0.2666667) *
##            35) AGE>=32.5 28  13 NB (0.4642857 0.5357143)
##              70) MARSTAT=single 7   2 B (0.7142857 0.2857143) *
##              71) MARSTAT=not single 21   8 NB (0.3809524 0.6190476)
##               142) NUMHH>=3.5 10   4 B (0.6000000 0.4000000) *
##               143) NUMHH< 3.5 11   2 NB (0.1818182 0.8181818) *
##         9) AGE>=68 14   5 NB (0.3571429 0.6428571) *
##       5) AGE< 28.5 17   6 NB (0.3529412 0.6470588) *
##     3) INCOME< 23500 60  19 NB (0.3166667 0.6833333)
##       6) EDUCATION>=12.5 27  13 NB (0.4814815 0.5185185)
##        12) MARSTAT=not single 11   2 B (0.8181818 0.1818182) *
##        13) MARSTAT=single 16   4 NB (0.2500000 0.7500000) *
##       7) EDUCATION< 12.5 33   6 NB (0.1818182 0.8181818) *
```

## The tree



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

```
##     pred.test
##       B  NB
##   B  89 18
##   NB 47 46
```

```
sum(diag(t))/sum(t)
```

```
## [1] 0.675
```

## MARS: Multivariate Adaptive Regression Splines

- MARS is an adaptive procedure for regression, and is well suited for highdimensional 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 method to improve the latter's performance in the regression setting.
- 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 suface
- At each step it is chosen a couple of basis functions recursively selecting the variable $X$ that is most appropriate and the position of the knot.

## MARS

- MARS builds models of the form

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

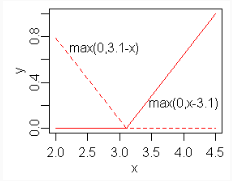
- The model is a weighted sum of basis functions $B_i(x)$. Each $c_i$ is a constant coefficient.
- Each basis function $B_i(x)$ takes one of the following three forms: 1.a constant

  **2.** a hinge function. A hinge function has the form
  $\max(0, x - \mathit{const})$ or $\max(0, \mathit{const} - x)$.
  MARS automatically selects variables and values of those variables for knots of the hinge functions.

  **3.** a product of two or more hinge functions. These basis functions can model interaction between two or more variables.

## MARS

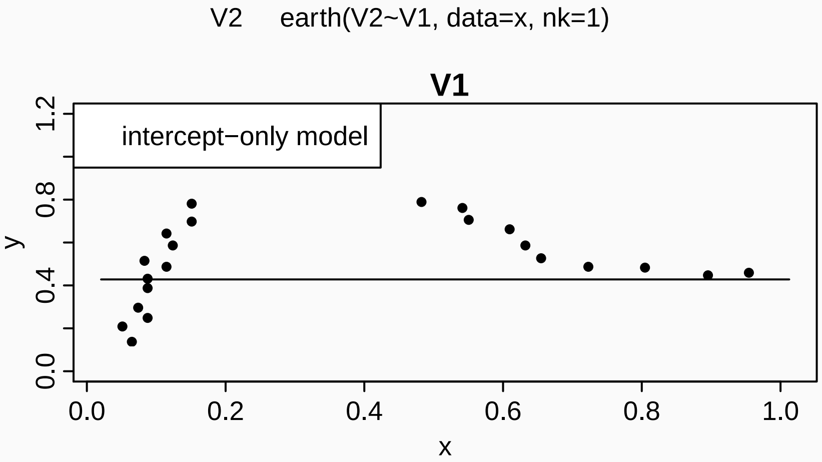
This is an example of a couple of Hinge functions



- 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", ylim=c(0,1.2)); points(x,pch=20)
```



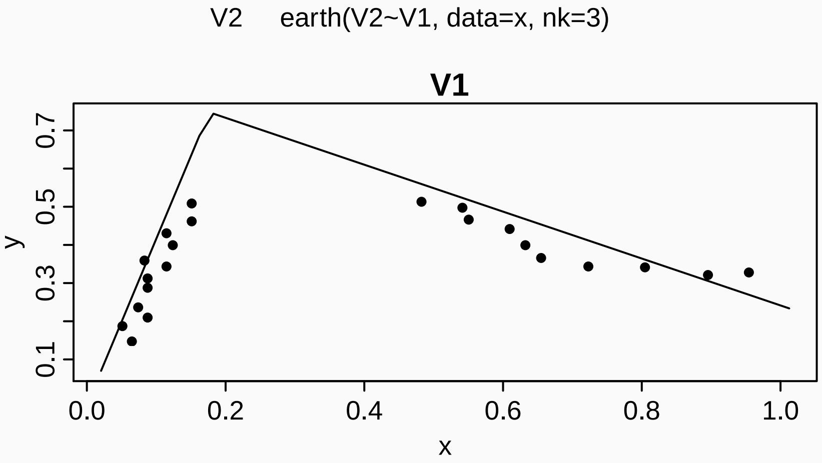
## An example

```r
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

```r
mod2=earth(V2~V1,data=x,nk=3)
plotmo(mod2,xlab="x",ylab="y"); points(x,pch=20)
```



## An example

```r
summary(mod2)
```

```
## Call: earth(formula=V2~V1, data=x, nk=3)
##
##                 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 3
## 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
```

## Ensemble methods

## Combining multiple predictions: Model averaging

- Classification trees can be simple, but often produce noisy (bushy) and weak classifiers
  - **Bagging** *(Breiman, 1996)*: Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote
  - **Boosting** *(Freund & Shapire, 1996)*: Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote
  - **Random Forests** *(Breiman 1999)*: Fancier version of bagging.
- Note however that the idea of combining multiple predictions or classifications can be used for any technique (i.e., logistic classification, NN, etc.) and it is not limited to trees
- This idea is closely related with model averaging: a strategy for model selection and evaluation of unvertainty in Bayesian analyses

## Combining predictions (classifications)

- The idea is to combine the output of different learners for each data point ($y$, $\boldsymbol{x}$). This help when learners have complementary strengths.
- Suppose training data are available in the form of the $p$ covariates $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ and the response is (target) is $y$ . Let $\hat{y}_1 = f_1(\boldsymbol{x}), \ldots, \hat{y}_M = f_1(\boldsymbol{x})$ be $M$ different predictions (estimates, "experts evaluations") for the same data point.
- A simple combined vote takes their average

$$f_{comb}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{x})$$

In the classification setting for each class $k$ we have a prediction $f_m^k(x, t)$ equal to 0 or 1. Then

$$f_{comb}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m^k(\boldsymbol{x})$$

for each class $k$ and $f_{comb}^k(x, t)$ is the proportion of votes for class $k$. We predict the class with the most number of votes **(majority vote)**.

## Bagging

- Bagging or bootstrap aggregation averages a given procedure over many samples, to reduce its variance
- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
- Instead, we can
  - bootstrap, by taking repeated samples from the same training data set
  - use the $b$-th bootstrapped training set to get the prediction $\hat{f}^b(x)$ and
  - average all the predictions, to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(\boldsymbol{x})$$

- This is called bagging. In classification problems it uses majority votes.
- Bagging can dramatically reduce the variance of unstable procedure (like trees), leading to improved prediction.
- Bagging averages many trees, produces smoother decision boundaries, reduces the variance, but can slightly increase bias

## Out-of-bag

- Using random samples of observations allows the use of the out-of-bag tool, for easy estimation of prediction errors.
- In each bootstrap sample, some of the data of the original training set are excluded.
- On average, each bagged sample makes use of around two-thirds of the observations
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations
- For each classifier $\hat{f}^b(\boldsymbol{x})$ the data of training set that are not in sample can be used as a test set. This will give $B/3$ predictions for the $i$-th observation.
- Estimate the misclassification error on these data outside the sample used for the fit (out-of-bag), so avoiding cross-classification for large data sets

## Random Forest

- A quite popular refinement of bagging. Particularly when bagging trees for which was originally developed. We will describe this version.
- At each tree split, a random sample of $m$ features is drawn, and only those $m$ features are considered for splitting.
- Typically $m = \sqrt{p}$ or $\log_2 p$, where $p$ is the number of features (covariates)
- For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored (out-of-bag)
- Random forests tries to improve on bagging by “de-correlating” the trees, and reduce the variance.
- Each tree has the same expectation, so increasing the number of trees does not alter the bias of bagging or random forests.

## Variable importance

Random forests can be used to rank the importance of variables in a regression or classification problem.

- For each tree grown in a random forest, calculate number of votes for the correct class in out-of-bag data.
- Now perform random permutation (shuffling) of a predictor's values (let's say variable-k) in the OOB data and then check the number of votes for correct class.
- Subtract the number of votes for the correct class in the variable-k-permuted data from the number of votes for the correct class in the original OOB data.
- The average of this number over all trees in the forest is the raw importance score for variable k. The score is normalized by taking the standard deviation.
- Variables having large values for this score are ranked as more important. It is because if building a current model without original values of a variable gives worse prediction, it means the variable is important.

## Boosting

- Designed, initially, exclusively for classification problems
- Idea: Like bagging, but take unequal probability bootstrap samples. Put more weight on observations that are misclassified, to make the classifier work harder on those points. Invention of Freund e Schapire (1997)
- Details
    - Start with equal observation weights $p_i = 1/n$
    - At iteration $t$, draw a bootstrap sample with the current probabilities $p_1, p_2, \ldots, p_n$, compute the classifier and $e_t$ , the error rate of the classifier on the original sample. Let $\beta_t = e_t/(1 - e_t)$
    - For those points that are classified correctly, decrease their probabilities $p_i = p_i\beta_t$ and normalize them
    - Do this for many (say 1000) iterations.
- At the end, take a weighted vote of the classifications, with weights $\alpha_t = log(1/\beta_t)$ (more weight on classifiers with lower error).
- Boosting can improve bagging in many instances

Weighting decorrelates the trees, and focuses on regions missed by past trees.