Classification And Regression Trees

Mathias Bourel

IMERL - Facultad de Ingeniería, Universidad de la República, Uruguay

April 3, 2019

Framework of Machine Learning

General framework:
$\mathcal{L}$ a data basis.

Framework of Machine Learning

General framework:
$\mathcal{L}$ a data basis. We search about $f : \mathcal{X} \to \mathcal{Y}$ a good predictor or a good explainer.

- Supervised Learning: $\mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$

  X: input variable, independent variable, explanatory (real o multidimensional), continuous, categorical, binary, ordinal.

  Y: output variable, dependent variable, real o categorical.

  - Classification: $y \in \{-1, 1\}$ (binary) or $y \in \{1, \ldots, K\}$ (multiclass).
  - Regression: $y \in \mathbb{R}$.

- Unsupervised Learning $\mathcal{L} = \{x_1, \ldots, x_n\} \subset \mathcal{X} \subset \mathbb{R}^d$

  - Clustering
  - Density estimation

  In all cases, the sample $\mathcal{L}$ is a collection of n independent realizations of a multivariate random variable (X, Y) or X

Plan

Classification and Regression Trees

Figure: Construcción geométrica. Joaquin Torres García (1929)

Classification and Regression Trees

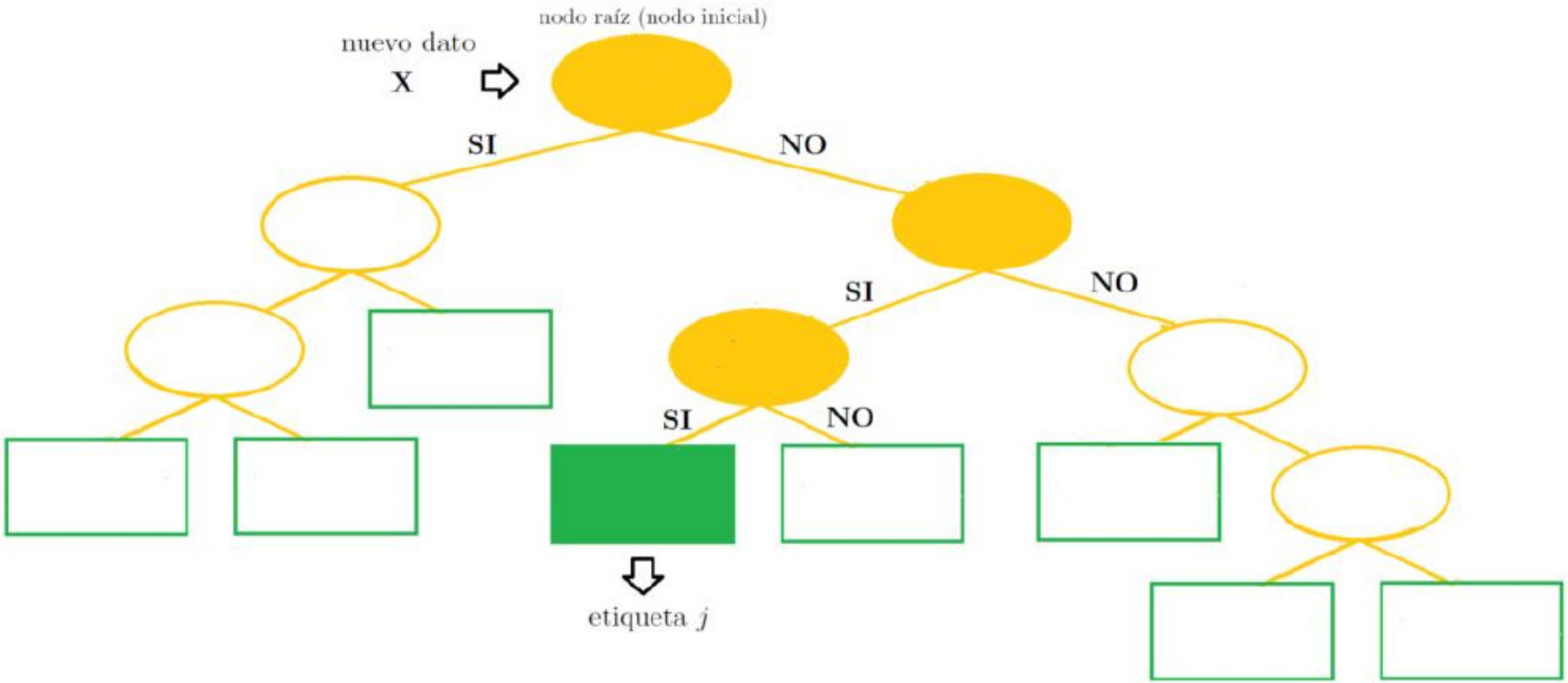
- It is a Machine Learning method.
- The idea of binary trees consists of recursively dividing the data set until reaching k terminal nodes (sheets)
- Explanatory variables can be quantitative or qualitative.
- At each stage of the algorithm, the best rule that divides the node into two is sought, in the most homogeneous way possible. This rule is of the type:

$$X_i \leq c \quad \text{vs} \quad X_i > c \text{ if } X_i \text{ is quantitative}$$

$$X_i \in \mathcal{A} \quad \text{vs} \quad X_i \in \mathcal{A} \text{ if } X_i \text{ is categorical}$$

- It is sought in each division to reduce the impurity of the parent node when its two children nodes appear.
- A stopping criterion is needed: for example minimum quantity of observations in the leaves or threshold on the criterion of impurity.

Classification and Regression Trees



Classification and Regression Trees

The construction of a tree requires defining:

- A partition criterion: how to perform binary subdivisions.
- A stop criterion: to consider when a node is considered terminal and the process is stopped.
- An assignment criterion: for the assignment of the label to each sheet.

But it is the same principle in classification and in regression (this fact is different for another method like SVM for example)

Classification and Regression Trees

A partition of space X is found and we assign a value (regression problem) or a category (classification problem) at each elements of the partition. This can be written linearly as

$$\mathbb{E}(Y|X=x) = \sum_{i=1}^{q} c_j \mathbb{1}_{N_j}(x)$$

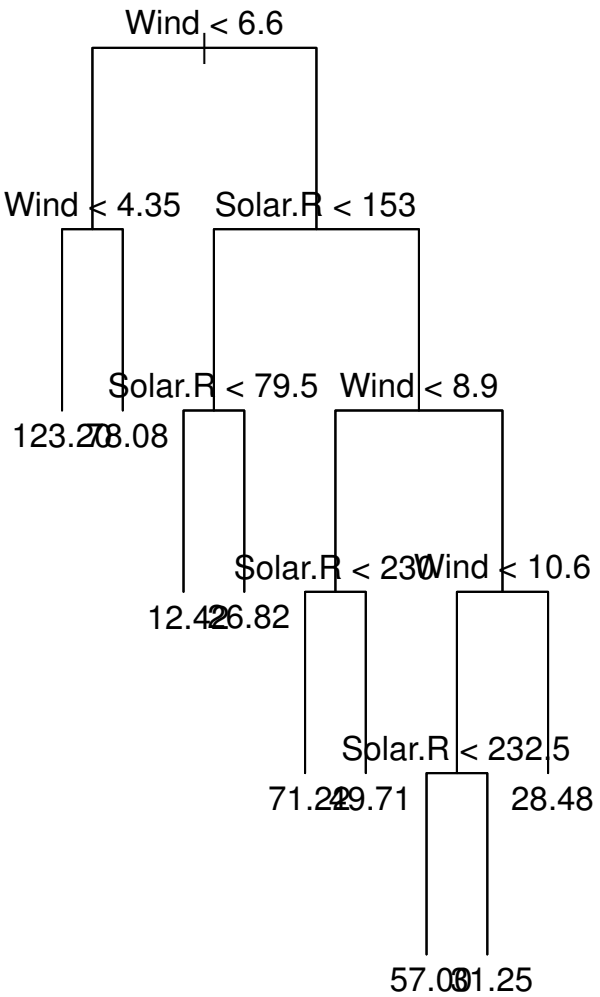
where

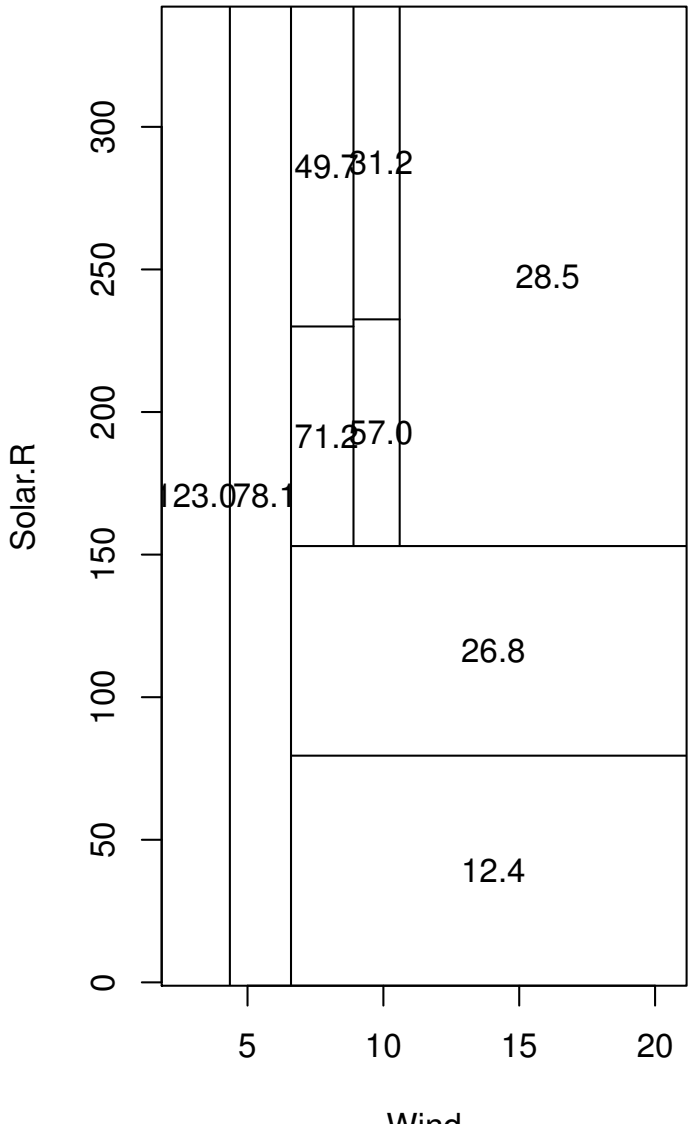
$$\widehat{c}_j = \frac{\sum_{i:X_i \in N_j} Y_i}{\#N_j} \quad \text{(regression)}$$

$$\widehat{c}_j = \text{majority class in } N_j \quad \text{(classification)}$$

Class: piecewise constant functions

Classification and Regression Trees

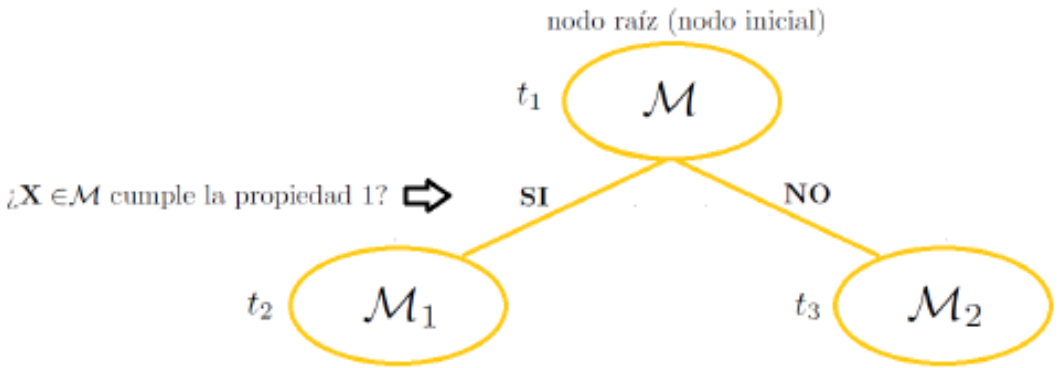




Partition criteria

All observations are in a root node. By means of a criterion of partition (criterion that involves the characteristics of the observations) this root is divided into two sub-samples, that is, two child nodes so that the children are more homogeneous in relation to Y that the parent node (decrease in impurity). And the process is repeated again.

A node is pure or homogeneous if it contains a single class. Otherwise is impure or heterogeneous.



Partition criteria (classification)

An impurity function $\phi : \{p = (p_1, \ldots, p_K) \in \mathbb{R}^K : \ p_i \geq 0, \sum_{i=1}^{K} p_i = 1\} \to \mathbb{R}$ must:

- be symmetric (that is, if the p_j is swapped, the function does not change).
- have minimums in the canonical basis of $\mathbb{R}^K$.
- its only maximum is $\left(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\right)$

Example of impurity functions:

1. $\phi(p) = 1 - ||p||_\infty = 1 - \max_k \{p_1, \ldots, p_K\}$ (classification error)
2. $\phi(p) = -\sum_{k=1}^{K} p_k \log(p_k)$ (entropy)
3. $\phi(p) = 1 - \sum_{k=1}^{K} p_k^2 = \sum_{k=1}^{K} p_k(1 - p_k) = \sum_{k \neq k'} p_k p_{k'}$ (Gini Index)

Note that:

- The entropy of a node with a single class is zero, because the probability is one and $\log(1) = 0$ (log is in base 2). Entropy reaches maximum ($\log(K)$) value when all classes have the same probability.
- Gini index of a node with a single class is zero. Gini index reaches maximum $(1 - 1/K)$ value when all classes have the same probability.
- Classification error of a node with a single class is zero. Classification error reaches maximum $(1 - 1/K)$ value when all classes have the same probability.

Gini index

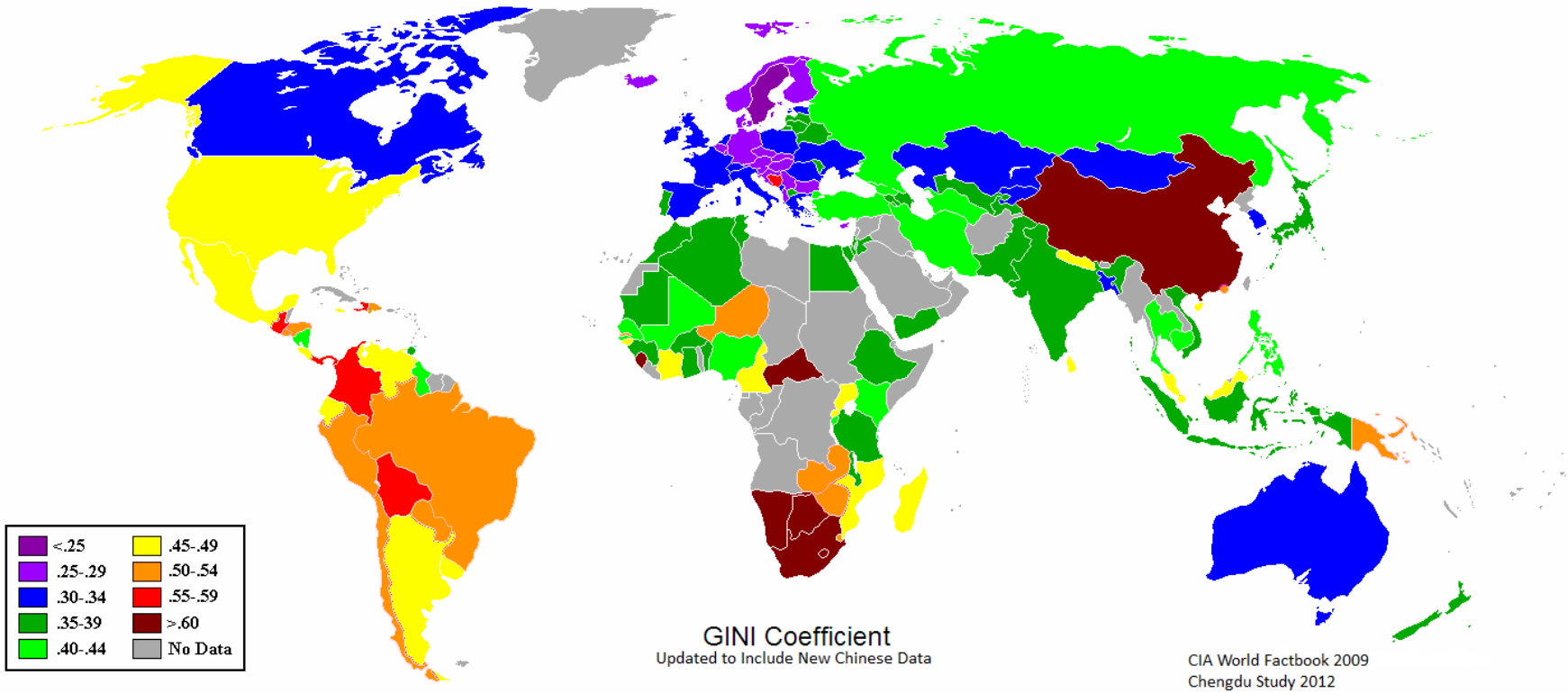


Figure: Gini coefficient map 2009

For a discrete probability distribution with probability mass function p_i, $i = 1, \dots n$, where p_i is the fraction of the population with income or wealth $y_i > 0$, the Gini coefficient is

$$G = \frac{1}{2\mu} \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_j |y_i - y_j| \text{ where } \mu = \sum_{i=1}^{n} y_i p_i.$$

Partition criteria (classification)

For example, assume we have a database with 100 observations: 40 are red, 30 are blue and 30 are green. Based on these data we can compute probability of each class. Since probability is equal to frequency relative:

$$\mathbb{P}(\text{red}) = \frac{40}{100} = 0,4 \quad \mathbb{P}(\text{blue}) = \frac{30}{100} = 0,3 \quad \mathbb{P}(\text{green}) = \frac{30}{100} = 0,3$$

- the entropy is $-0,4 \times \log(0,4) - 0,3 \times \log(0,3) - 0,3 \times \log(0,3) = 1,571$
- the gini index is $1 - (0,4^2 + 0,3^2 + 0,3^2) = 0,660$
- the classification error is $1 - 0,4 = 0,6$

Partition criteria (classification) Node impurity

If $N(t)$ is the quantity of observations of $\mathcal{L}$ that belong in node t and $N_k(t)$ is the number of observations in t that have label k ($k \in \{1, \ldots, K\}$), then the probability that an observation of t belongs to class k is

$$p_k(t) = \frac{N_k(t)}{N(t)}$$

If ϕ an impurity function, the node's impurity of t is defined as:

$$i(t) = \phi\big(p_1(t), p_2(t), \ldots, p_K(t)\big)$$

For example if $\phi = 1 - ||p||_\infty$ then

$$i(t) = 1 - \max_k \{p_1(t), p_2(t), \ldots, p_K(t)\} = 1 - \max_k \left\{ \frac{N_1(t)}{N(t)}, \frac{N_2(t)}{N(t)}, \ldots, \frac{N_K(t)}{N(t)} \right\}$$

$$i(t) = \frac{N(t) - N_{j^*}(t)}{N(t)} = \frac{\text{misclassified in } t}{N(t)}$$

where j^* the majority class in t.

Partition criteria (classification)

The Gini index and the entropy are more sensitive to changes in the probabilities of the nodes than the classification error (the latter may have many ties).

The Gini index $\sum_{k=1}^{K} p_k(1-p_k)$ is a measure of the total variation on the K classes. If all probabilities are close to 0 or 1, the Gini index is low (+ purity). Idem for entropy.

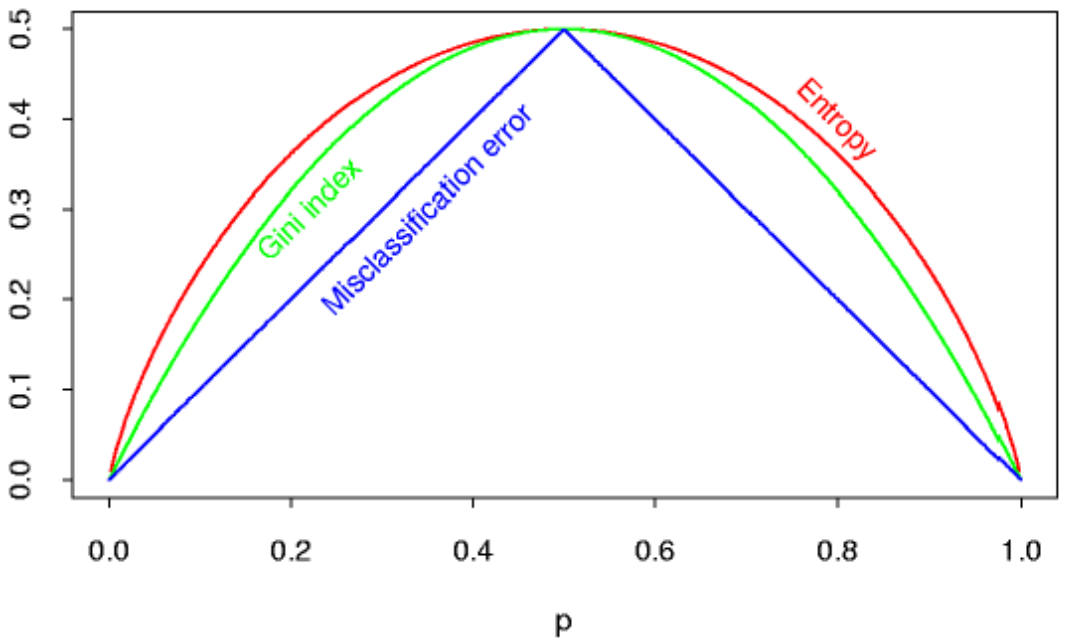


Figure: Hastie et al. (2001)

Partition criteria (classification)

The impurity variation of node t respect to its two children t_L and t_R when performing the s e partitions:

$$\Delta\, i(t,s) = i(t) - p_L i(t_L) - p_R i(t_R) \geq 0$$

$$\Delta\, i(t,s) = i(t) - \frac{N(t_L)}{N(t)} i(t_L) - \frac{N(t_R)}{N(t)} i(t_R)$$

For example, if the impurity function is the classification error, then

$$\Delta\, i(t,s) = \frac{\text{misclassified in } t - \text{misclassified in } t_L - \text{misclassified in } t_R}{N(t)}$$

We choose then within all possible partitions $\mathcal{S}_t$ of t, on values and characteristic variables, the one that verifies that

$$s^*(t) = \underset{s \in \mathcal{S}_t}{\text{Argmax}}\, \Delta\, i(t,s)$$

Partition criteria (classification)

The classification error is the same for the two trees but the Gini index and entropy are defined by the tree with a pure terminal node (exercise).

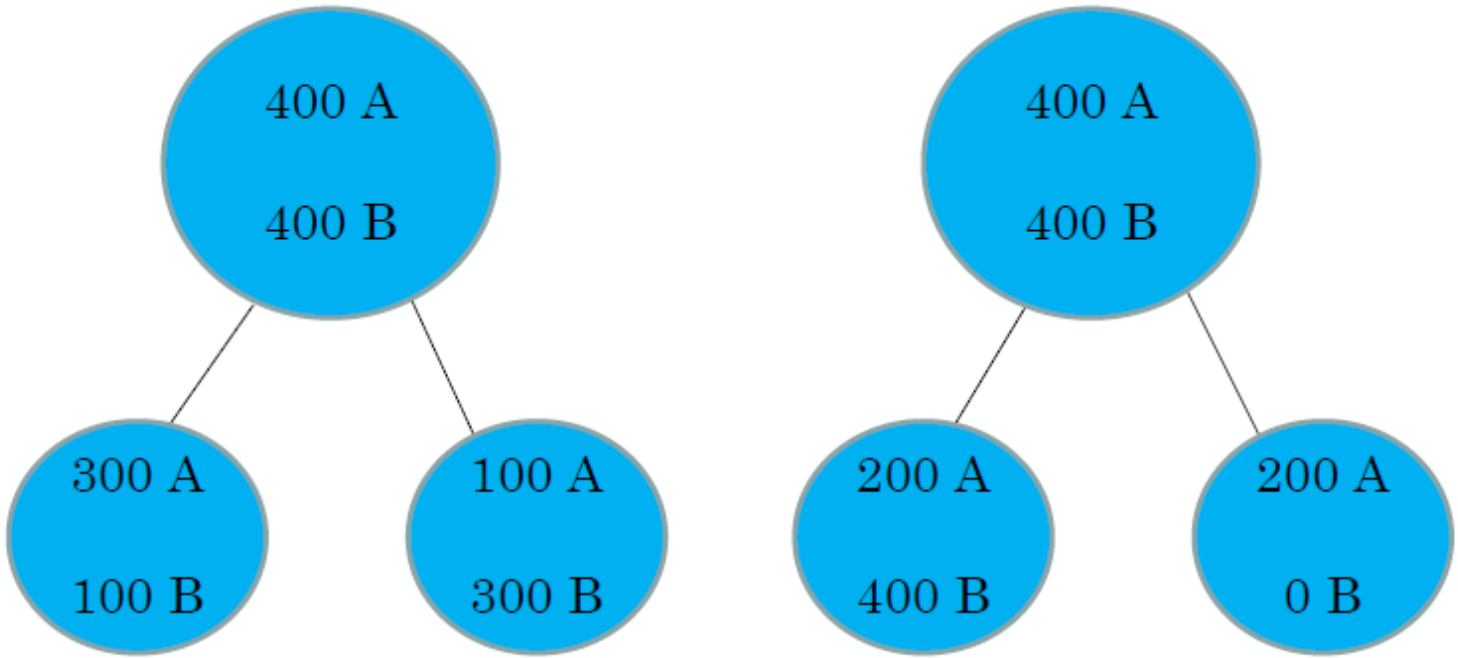


Figure: Breiman et. al, [1]

Partition criteria (classification)

The global impurity of the tree T is

$$I(T) = \sum_{t \in \tilde{T}} \underbrace{p(t)i(t)}_{R(t)}$$

where $\tilde{T}$ is the set of leaves of T, $p(t)$ is the probability of belonging to the node t and $i(t)$ is the impurity of t.

In [1], Breiman et. they prove that maximizing the impurity difference in each node is equivalent to minimizing the global impurity of the tree.

Partition criteria (regression)

In regression, deviance in the node is used to measure heterogeneity of a node

$$R(t) = \frac{1}{N} \sum_{X_i \in t} (Y_i - \overline{Y}(t))^2$$

where $\overline{Y}(t) = \frac{1}{\#t} \sum_{X_i \in t} Y_i$.

Observe that $R(t) = \frac{1}{N} \sum_{X_i \in t} (Y_i - \overline{Y}(t))^2 = \frac{\#t}{N} \frac{1}{\#t} \sum_{X_i \in t} (Y_i - \overline{Y}(t))^2 = p(t)\text{var}(t)$

We choose then within all possible partitions $\mathcal{S}_t$ of t, on values and characteristic variables, the one that verifies that minimizae the internal variance after the split

$$s^*(t) = \underset{s \in \mathcal{S}}{\text{Argmax}}\, \Delta\, R(t, s)$$

where

$$\Delta\, R(t, s) = R(t) - R(t_L) - R(t_R) \geq 0$$

Stop criterion

If i is the classification error $i(T) = R(T)$ is the estimation of the classification error and look for the tree that has smaller global impurity equivalent to the one that has $R(T)$ smaller and this will be T_{max}.
The stop criterion is defined by the user beforehand. It must be chosen so that the tree is not too large on the one hand and does not conform too much to the sample from which the tree develops. There are two main criteria:

1. Choose a threshold from which we decide that a node is pure, that is, a β such that if $i(t) \geq \beta$ we continue with the partition of t an if $i(t) < \beta$ we stop partition in t.

   If β is very small, this increases the complexity of the tree since the number of leaves can be close to N (the size of the sample) and we lose in generalization (one sheet for each observation).
2. Decide that a node does not divide more if it contains less than m observations.

Classification and Regression Trees

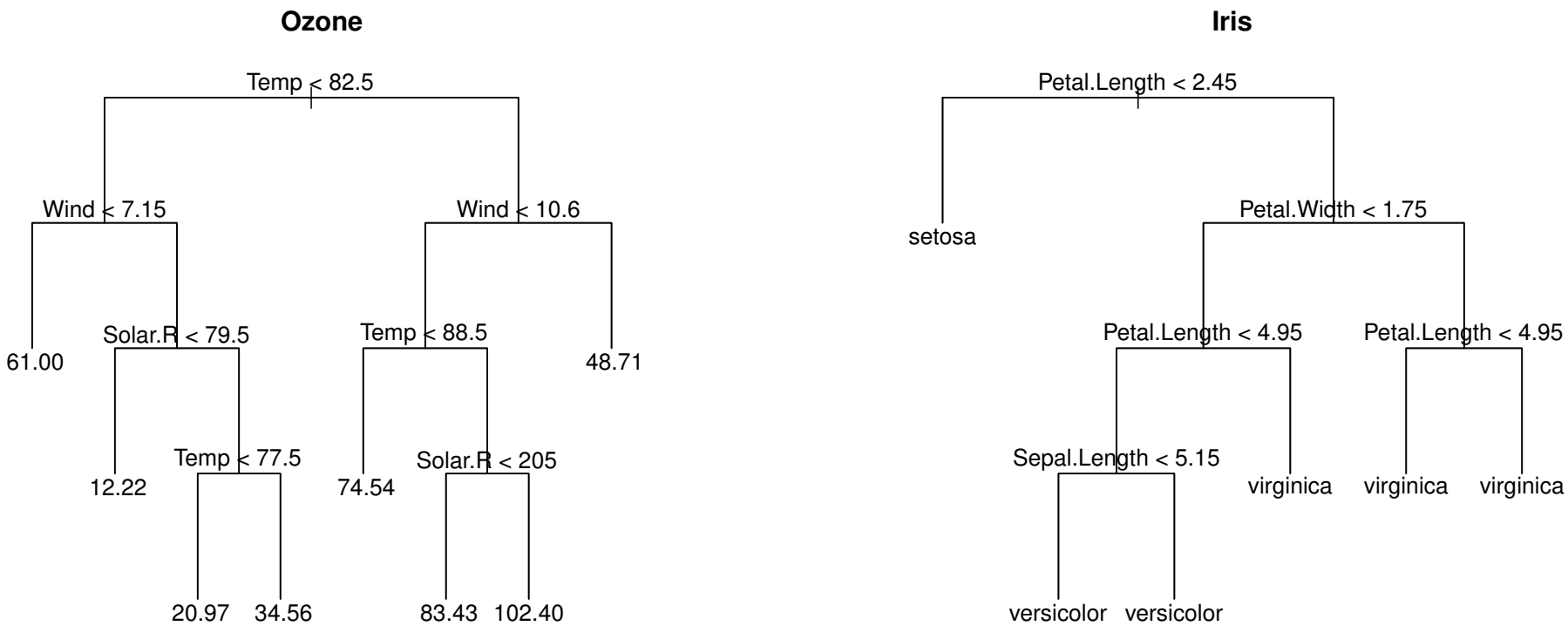
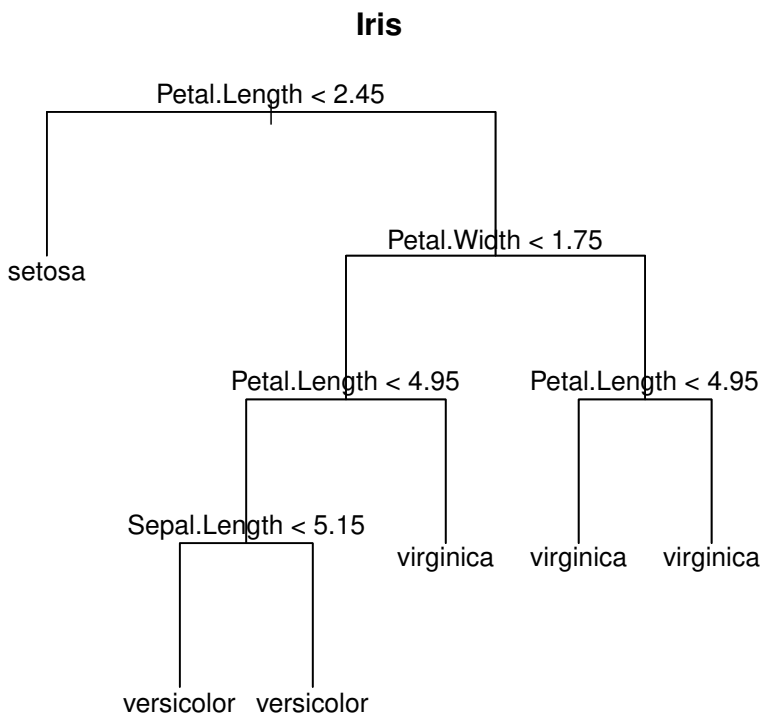


Figure: Classification and Regression Trees

Classification and Regression Trees



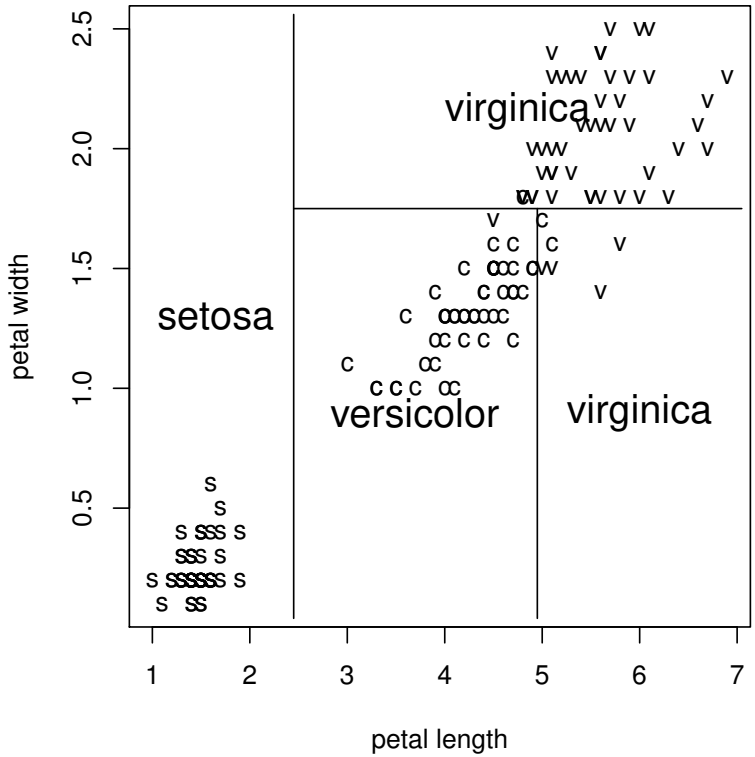


Figure: Classification and Regression Trees

Assignment criteria

In a terminal node:

- For classification:

  The class that is most represented in each terminal node is chosen (simple majority vote. If the maximum is reached for two or more classes, this class is assigned randomly.

- For regression:

  The average of the values of the dependent variable in the leaf

Surrogate Splits

A surrogate split partitions the data in a way that is as close as possible to the primary split (the variable whose makes the best partition). Generally, these surrogates have two major purposes:

1. enable flexibility in terms of missing data,

Surrogate Splits

A surrogate split partitions the data in a way that is as close as possible to the primary split (the variable whose makes the best partition). Generally, these surrogates have two major purposes:

1. enable flexibility in terms of missing data,

   In the case of missing data, any observation that is missing in the split variable can be classified using the first surrogate variable, if available, and if not available, the second surrogate, and so forth.

2. reveal aspects of variable importance in the data.

Surrogate Splits

A surrogate split partitions the data in a way that is as close as possible to the primary split (the variable whose makes the best partition). Generally, these surrogates have two major purposes:

1. enable flexibility in terms of missing data,

   In the case of missing data, any observation that is missing in the split variable can be classified using the first surrogate variable, if available, and if not available, the second surrogate, and so forth.

2. reveal aspects of variable importance in the data.

   Surrogate variables also play a role in variable importance. In addition to the structural interpretation of the tree itself, the relative importance of each variable is often assessed using a variable importance measure. The calculation of variable importance in the rpart package is performed over surrogate splits: it is the sum of the goodness of split measures for each split for which the variable was in the role splitting in the tree, and the fit for all splits in which it was a surrogate. These importance measures are scaled to sum to 100 and then rounded. Variables that are considered negligible are omitted. From this point of view, a variable that does not necessarily enter the tree, may be considered important based on its variable importance measure - driven by surrogate splits.

https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

Surrogate Splits

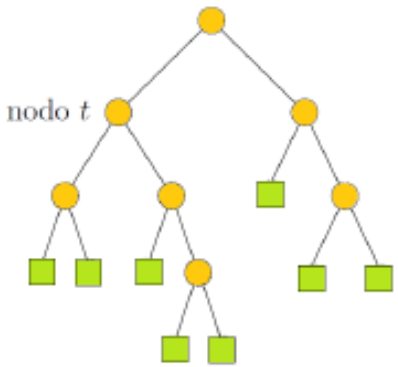
```
Variable importance
      charDollar           remove           num000            money  charExclamation      capitalLong
              21               13                8                7                5                5
   capitalTotal           credit            order          receive      capitalAve               hp
               5                4                3                3                3                3
       addresses         business         internet           people         charHash             free
               3                2                2                2                2                1
             hpl             over             your charRoundbracket              our              you
               1                1                1                1                1                1
          telnet
               1

Node number 1: 4601 observations,    complexity param=0.4765582
  predicted class=nonspam  expected loss=0.3940448  P(node) =1
    class counts:  2788  1813
   probabilities: 0.606 0.394
  left son=2 (3471 obs) right son=3 (1130 obs)
  Primary splits:
      charDollar      < 0.0555 to the left,  improve=714.1697, (0 missing)
      charExclamation < 0.0795 to the left,  improve=711.9638, (0 missing)
      remove          < 0.01   to the left,  improve=597.8504, (0 missing)
      free            < 0.095  to the left,  improve=559.6634, (0 missing)
      your            < 0.605  to the left,  improve=543.2496, (0 missing)
  Surrogate splits:
      num000       < 0.055  to the left,  agree=0.839, adj=0.346, (0 split)
      money        < 0.045  to the left,  agree=0.833, adj=0.321, (0 split)
      credit       < 0.025  to the left,  agree=0.796, adj=0.169, (0 split)
      capitalLong  < 71.5   to the left,  agree=0.793, adj=0.158, (0 split)
      order        < 0.18   to the left,  agree=0.792, adj=0.155, (0 split)
      capitalTotal < 693.5  to the left,  agree=0.790, adj=0.143, (0 split)
      receive      < 0.035  to the left,  agree=0.789, adj=0.140, (0 split)
      remove       < 0.01   to the left,  agree=0.785, adj=0.125, (0 split)
      addresses    < 0.025  to the left,  agree=0.785, adj=0.124, (0 split)
      internet     < 0.035  to the left,  agree=0.777, adj=0.093, (0 split)
      business     < 0.065  to the left,  agree=0.777, adj=0.091, (0 split)
      people       < 0.155  to the left,  agree=0.775, adj=0.086, (0 split)
      capitalAve   < 5.8895 to the left,  agree=0.775, adj=0.086, (0 split)
      charHash     < 0.0075 to the left,  agree=0.771, adj=0.067, (0 split)
      over         < 0.145  to the left,  agree=0.768, adj=0.054, (0 split)
```

Plan

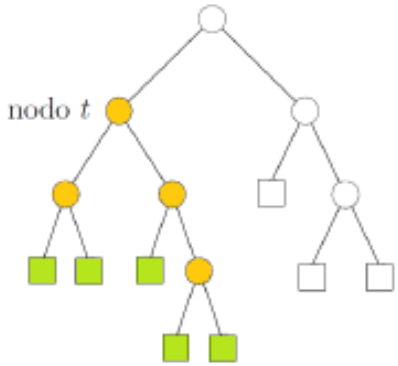
Pruning algorithm

Let t a node of T and we call branch coming form t to the subtree T_t of T that have t. The pruning of branch T_t consists in suppressing all the descendant nodes of t (except t). The tree obtained is noted by $T - T_t$. If T' is obtained from T by successive pruning of branches, we say that T' is a subtree of T and we note

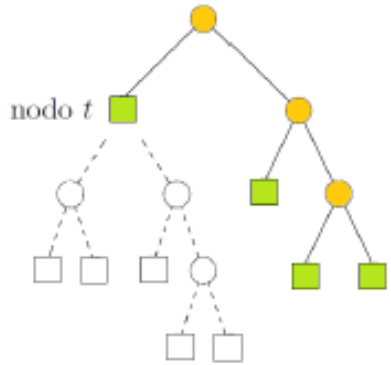
$$T' < T$$



árbol T



rama T_t



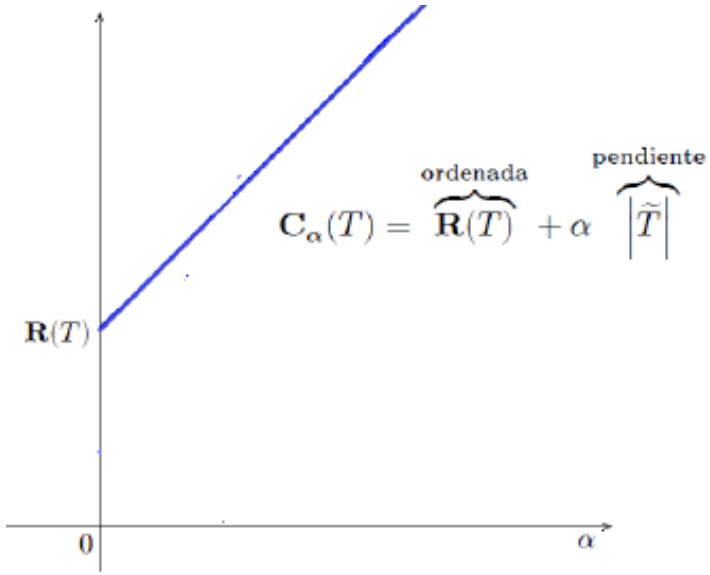
árbol podado $T - T$

Pruning algorithm

So let's take into account, in addition to the classification error of T, its complexity measured by $|\tilde{T}|$ (the number of sheets). It is a trade off of combining good classification and simplicity of the classifier.

Let $\alpha \geq 0$. The cost-complexity measure of parameter α associated with tree T is:

$$C_\alpha(T) = R(T) + \alpha|\tilde{T}| \quad \text{(function of } \alpha\text{)}$$



where $R(T)$ is the classification error and $\tilde{T}$ the complexity of T (number of leaves).
Big values of α will penalize trees with many leaves, while small values of α will give little importance to the size. In the case that $\alpha = 0$ we are left with the maximal tree that minimizes the error. As we increase the value of α the size will be penalized, and then we get trees that are getting smaller and smaller but with a big error.

Pruning algorithm

In regression the cost-complexity measure is

$$C(T) = \sum_{t=1}^{|\tilde{T}|} \sum_{x_i \in t} (y_i - \hat{y}_t)^2 + \alpha |\tilde{T}|$$

(the first sum is over the leaves of T).

We return to the sequence of trees built together with their respective cost-complexity levels:

$$T_1 > T_2 > \cdots > \{t_1\} = T_K$$

$$0 = \alpha_1 < \alpha_2 < \cdots < \alpha_K$$

T_1 is more complex but has the less error and $\{t_1\} = T_K$ is more simple but have a big error.

¿How we choose the best of these subtrees with respect to our cost-complexity criterion?

Pruning algorithm

With a Test sample (large sample)

If the sample is sufficiently large, we choose the pruned tree dividing in two parts the sample: $\mathcal{L}_1$ to train and $\mathcal{L}_2$ to test ($\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$).
More precisely, with $\mathcal{L}_1$ we construct the sequence of tree and for all of them we compute

$$R(T_k) = \frac{\#\text{obs. misclassfied from } L_2 \text{ by } T_k}{\#\mathcal{L}_2} \quad \forall\, k = 1, \ldots, K$$

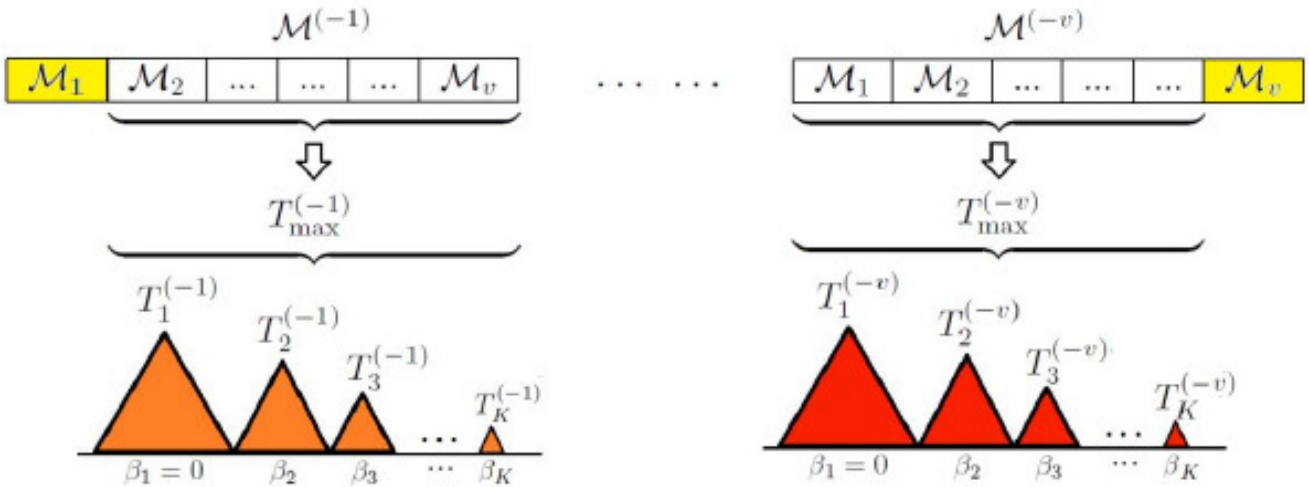
We select T_{k_0} of the original sequence such that

$$T_{k_0} = \underset{T_k}{\text{Argmin}} R(T_k)$$

Pruning algorithm

Doing Cross validation (with few data).

- Original sample $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_V$ is divided randomly in V parts. Define $\mathcal{M}^{(-v)} = \mathcal{M} \setminus \mathcal{M}_v$ par todo $v = 1, \ldots, V$
- For all $\mathcal{M}^{(-v)}$ we construct the associated maximal tree $T_{\max}^{(-v)}$.
- From tree $T_{\max}^{(-v)}$, for all $k = 0, 1, 2, \ldots, K-1$ let $\beta_k = \sqrt{\alpha_k \alpha_{k+1}}$ and construct the sequence of trees $T_k^{(-v)}$ where $T_k^{(-v)} < T_{\max}^{(-i)}$ and $T_k^{(-v)}$ is the best tree with cost-complexity parameter β_k.



Pruning algorithm

- Compute the classification error de $T_k^{(-v)}$ over $\mathcal{M}_v$

$$R(T_k^{(-v)}) = \frac{\#\text{misclassified of } \mathcal{M}_v \text{ by } T_k^{(-v)}}{\#\mathcal{M}_v}$$

and the classification error of the crossvalidation for all $k = 1, \ldots, K$:

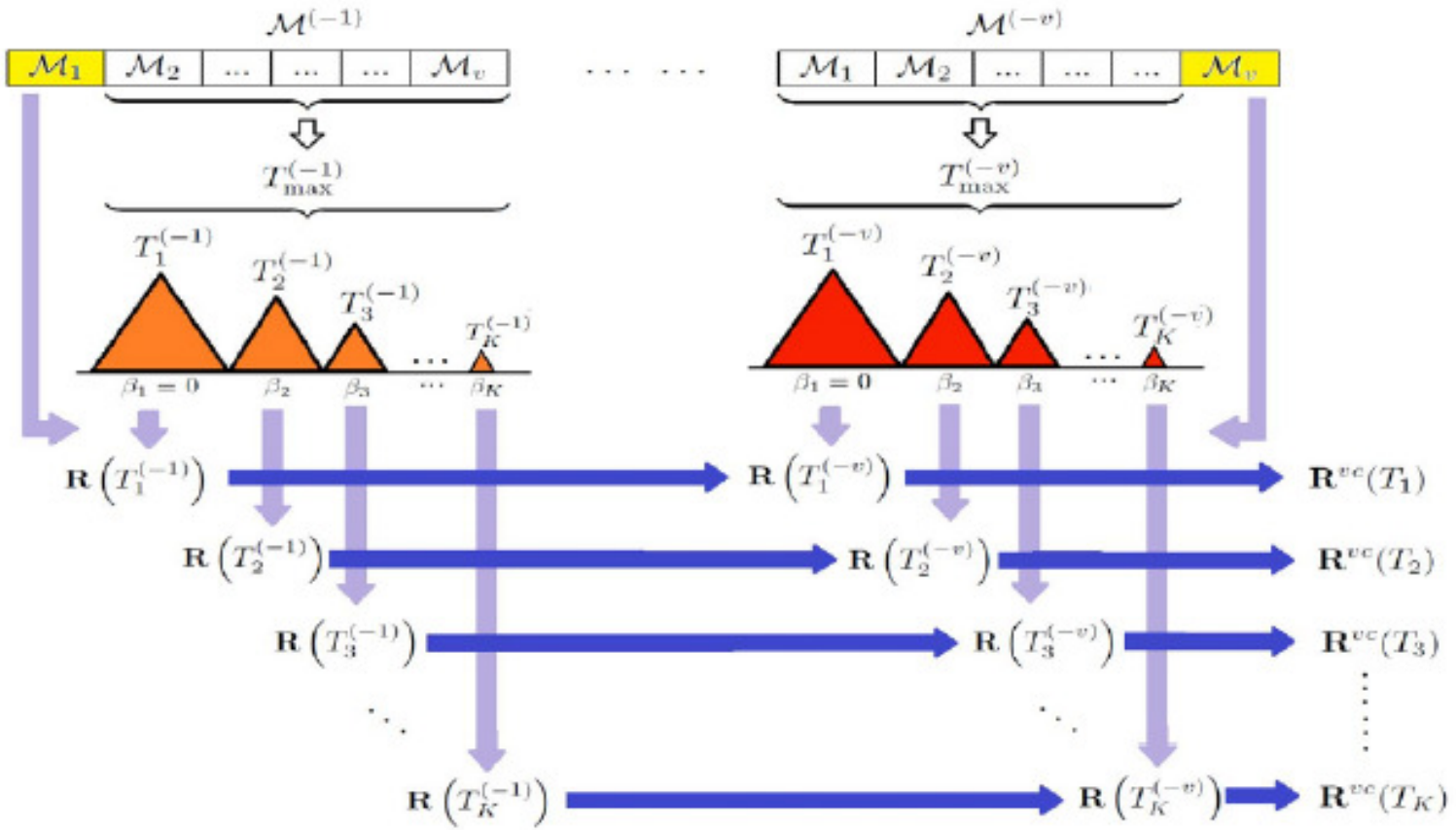
$$R^{vc}(T_k) = \frac{1}{V} \sum_{v=1}^{V} R\left(T_k^{(-v)}\right)$$

- We select from the original sequence tree T_{k_0} such that

$$T_{k_0} = \underset{T_k}{\text{Argmin}} R^{vc}(T_k)$$

Pruning algorithm

Scheme of the crossvalidation procedure:



We select, of the original sequence

$$T_{k_0} = \underset{T_k}{\text{Argmin}} R^{vc}(T_k)$$

(is the tree that on average presents the smallest error. Actually we select the best value of β_{k_0} that determines the "best" tree of the original sequence)

Pruning algorithm

By the 1-SE rule

The above procedure can be unstable in the sense that they can depend on how the initial participation is carried out both in the case of a test sample and cross-validation.

Pruning algorithm

By the 1-SE rule

The above procedure can be unstable in the sense that they can depend on how the initial participation is carried out both in the case of a test sample and cross-validation.

Breiman et al., (1984) suggest, instead of keeping the tree that minimizes the error according to the presented estimators, to retain the simplest tree whose error is less than the error of the tree with minimum error + its standard error (SE). The goal is to choose the most simple tree among all those who have a similar error.

1-SE rule

K	$card\left(\widetilde{A_K}\right)$	$R^{vc}(A_K) \pm SE$	$R^{res}(A_K)$	α
1	32	0.704 ± 0.060	0.145	0.000
8	16	0.639 ± 0.058	0.244	0.008
9	14	0.635 ± 0.058	0.276	0.008
10	12	0.632 ± 0.058	0.310	0.008
11*	11	0.603 ± 0.057	0.332	0.011
12**	8	0.634 ± 0.058	0.430	0.016
13	7	0.668 ± 0.059	0.464	0.017
14	5	0.687 ± 0.059	0.540	0.019
15	3	0.700 ± 0.058	0.619	0.020
16	2	0.729 ± 0.048	0.696	0.038
17	1	1.000 ± 0.000	1.000	0.152

Árbol 11: menor error por validación cruzada

Árbol 12: árbol más chico cuyo $R_{vc} < R_{vc}$ (árbol 11) + SE (árbol 11) = 0.660

Webb & Yohannes, 1999

Errors are normalized such that root node has error 1.

1-SE rule

K	$card\left(\widetilde{A_K}\right)$	$R^{vc}(A_K) \pm SE$	$R^{res}(A_K)$	α
1	32	0.704 ± 0.060	0.145	0.000
8	16	0.639 ± 0.058	0.244	0.008
9	14	0.635 ± 0.058	0.276	0.008
10	12	0.632 ± 0.058	0.310	0.008
11*	11	0.603 ± 0.057	0.332	0.011
12**	8	0.634 ± 0.058	0.430	0.016
13	7	0.668 ± 0.059	0.464	0.017
14	5	0.687 ± 0.059	0.540	0.019
15	3	0.700 ± 0.058	0.619	0.020
16	2	0.729 ± 0.048	0.696	0.038
17	1	1.000 ± 0.000	1.000	0.152

Árbol 11: menor error por validación cruzada

Árbol 12: árbol más chico cuyo $R_{vc} < R_{vc}$ (árbol 11) + SE (árbol 11) = 0.660

Webb & Yohannes, 1999

Errors are normalized such that root node has error 1.

In the table of the figure, a sequence of subtrees is shown with their respective complexity values, errors R^{vc} y R^{res} and complexity parameter on real data. In this case, T_{11} would be the tree chosen by the criterion of least cost-complexity, if we use the cross-validation estimator. Of all the trees that satisfy that their R^{vc} errors are less than $0,603 + 0,057 = 0,66$, we kept with the most simple, T_{12}, with 8 leaves.

1-SE rule

K	$card\left(\widetilde{A_K}\right)$	$R^{vc}(A_K) \pm SE$	$R^{res}(A_K)$	α
1	32	0.704 ± 0.060	0.145	0.000
8	16	0.639 ± 0.058	0.244	0.008
9	14	0.635 ± 0.058	0.276	0.008
10	12	0.632 ± 0.058	0.310	0.008
11*	11	0.603 ± 0.057	0.332	0.011
12**	8	0.634 ± 0.058	0.430	0.016
13	7	0.668 ± 0.059	0.464	0.017
14	5	0.687 ± 0.059	0.540	0.019
15	3	0.700 ± 0.058	0.619	0.020
16	2	0.729 ± 0.048	0.696	0.038
17	1	1.000 ± 0.000	1.000	0.152

Árbol 11: menor error por validación cruzada

Árbol 12: árbol más chico cuyo $R_{vc} < R_{vc}$ (árbol 11) + SE (árbol 11) = 0.660

Webb & Yohannes, 1999

Errors are normalized such that root node has error 1.

In the table of the figure, a sequence of subtrees is shown with their respective complexity values, errors R^{vc} y R^{res} and complexity parameter on real data. In this case, T_{11} would be the tree chosen by the criterion of least cost-complexity, if we use the cross-validation estimator. Of all the trees that satisfy that their R^{vc} errors are less than $0,603 + 0,057 = 0,66$, we kept with the most simple, T_{12}, with 8 leaves.

Observe that the classification error decreases as the complexity grows, the choice of the subtree by this criterion (R^{res}) will lead us to choose the maximal tree T_1, which is inconvenient as we said before, for being a model complex and over-adjusted to the training sample.

Plan

Adjustment measure of a tree

A global measure of adjustment is to look at the global deviance D. Deviance measures the difference in the fit between the model candidate and the saturated model (in the case of trees, the one with as many leaves as observations).

$$D = -2 \sum_{t \in \tilde{T}} \sum_{k} n_{tk} \log \hat{p}_{tk} \text{ (for classification)}$$

or

$$D = \sum_{t \in \tilde{T}} (1 - \sum_{k} \hat{p}_{tk}^2) \text{ (classification -this one is used by rpart-)}$$

where n_{tk} and $\hat{p}_{tk}$ are, respectively, the amount and proportion of observations of class k on the leaf t.

Adjustment measure of a tree

A global measure of adjustment is to look at the global deviance D. Deviance measures the difference in the fit between the model candidate and the saturated model (in the case of trees, the one with as many leaves as observations).

$$D = -2 \sum_{t \in \tilde{T}} \sum_{k} n_{tk} \log \hat{p}_{tk} \text{ (for classification)}$$

or

$$D = \sum_{t \in \tilde{T}} (1 - \sum_{k} \hat{p}_{tk}^2) \text{ (classification -this one is used by rpart-)}$$

where n_{tk} and $\hat{p}_{tk}$ are, respectively, the amount and proportion of observations of class k on the leaf t.

$$D = \sum_{t \in \tilde{T}} \sum_{x_i \in t} (y_i - \overline{y}_t)^2 \text{ (for regression)}$$

If the value of D is small, this indicates a good fit of the model to the training data.

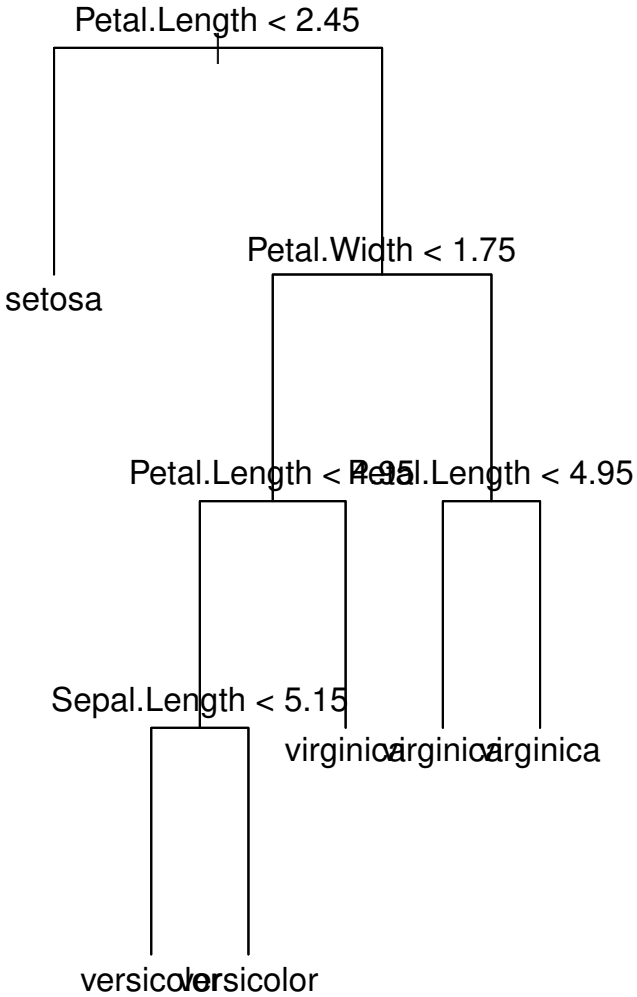
The average residual return (*residual mean deviance*) and pseudo-R^2 are defined as

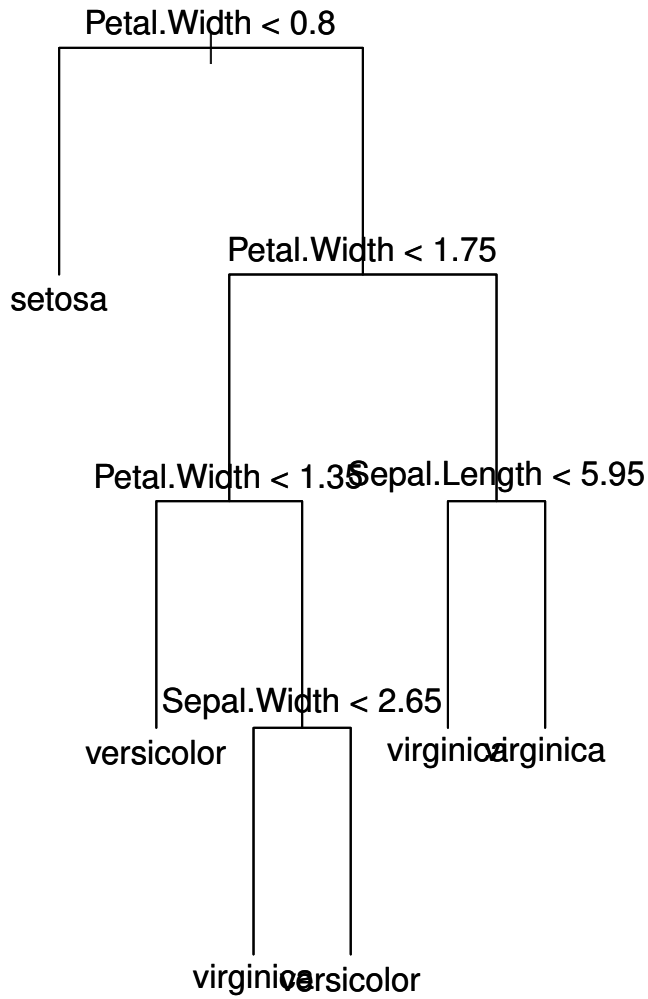
$$\frac{D}{N - |\tilde{T}|} \quad \text{and} \quad R^2 = \frac{D_{\text{root node}} - D}{D_{\text{root node}}}$$

Final considerations

- In [1] the consistency of CART is proved: if the number of observations is increased, the classification error of the model converges to the classification error.
- CART is easy to interpret.
- CART serves both for classification and for regression.
- CART performs well with missing data (surrogate variables).
- CART is an algorithm of the greedy type: it uses the best partition at every moment and therefore can leave aside variables that can be important in explaining variability of the data because they are highly correlated with variables that were used.
- CART is unstable: aggregation methods to stabilize it (Bagging, Boosting, Random Forest).

CART instability





CART vs LM

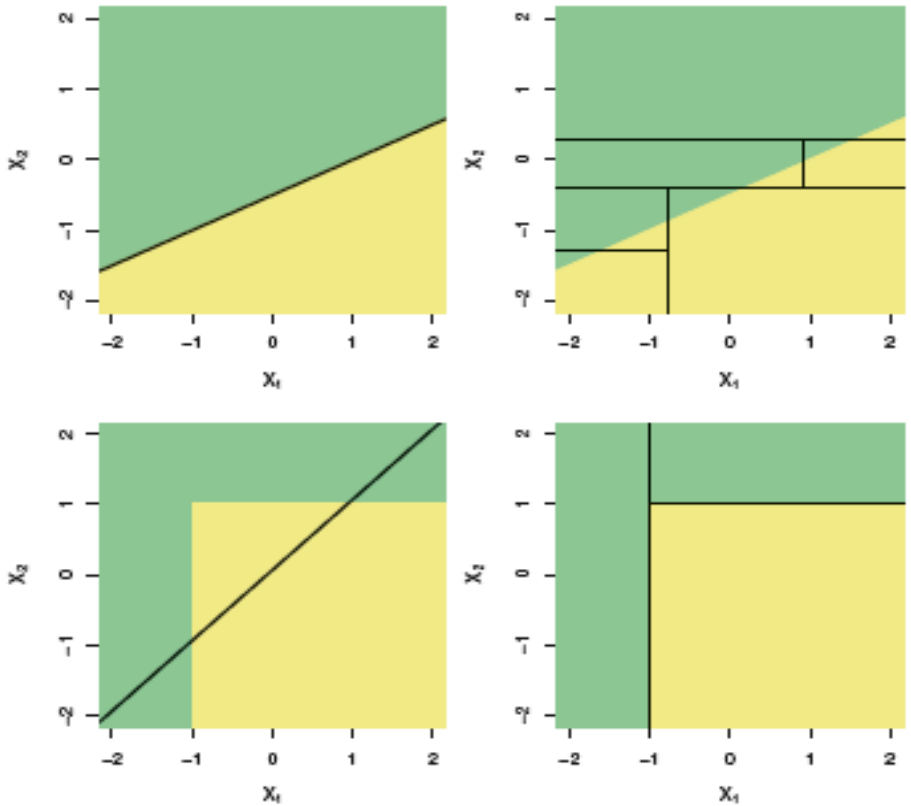


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Bibliography

1. Breiman, Freidman, Stone. Classification and Regression Trees, Chapman & Hall/CRC. 1984.
2. James, Witten, Hastie, Tibshirani. An introduction to Statistical Learning with application in R, Springer, 2013.
3. Hastie, Tibshirani, Friedman. The Elements of Statistical Learning, Springer, 2003.
4. Schapire, R.E and Freund, Y., *Boosting : Foundations and Algorithms*. Adaptive Computation and Machine Learning Series. Mit Press, 2012.
5. Freund, Y. and Schapire, E., A decision-theoretic generalization of on-line learning and application to boosting, *Journal of Computer and System Sciences*, 55(1): p 119-13, 1997.
6. Breiman, L., *Bagging predictors.*, Machine Learning 24, 123?140, 1996
7. Bourel, M., *Métodos de Agregación de modelos y aplicaciones*, Mémorias de trabajos de difusión científica y técnica, Vol. 10, p. 19-32, 2012.
8. Bourel, M., *Agrégation de modèles en apprentissage statistique pour l'estimation de la densité et la classification multiclasse*, Tesis de doctorado, Université Aix-Marseille, 2013.
9. Diaz, J., apuntes del curso 2013 de Aprendizaje Automático y Aplicaciones, FING, UdelaR.
10. Loh W.-Y. (2014) Fifty Years of Classification and Regression Trees, International Statistical Review, 82, pages 329348,