An Implementation of CART in Stata

Ricardo Mora

Universidad Carlos III de Madrid

Madrid, Oct 2015

Outline

Introduction

CART

- Tree-structured models are predictive models that use two-dimensional binary trees.
  - When the target variable can take a finite set of values, binary trees are called classification trees.
  - When the target variable can take continuous values (typically real numbers), they are called regression trees.
- Estimation of the tree is nontrivial when the structure of the tree is unknown: CART (Breiman et al, 1984)
  - CART: **C**lassification **a**nd **R**egression **T**rees
- Software packages: Salford Systems CART, Matlab, R
  - In Stata, module `<cart>` (Wim van Putten), performs CART analysis for failure time data.
- In this presentation, I first describe CART and then discuss its implementation with `<aries>`

Predictive learning

Predictive learning

- Consider the decomposition of output variable y between the effects of a set of observed controls x and that of all other factors such that

$$y = \mathrm{E}\left(y|x\right) + \epsilon$$

- The objective in predictive learning is to obtain a useful approximation of $\mathrm{E}\left(y|x\right)$
- Predictive learning is implemented through an optimization problem on a finite sample $\{y_i, x_i\}_i$ such as

$$\widehat{\mathrm{E}}\left(y|x\right) = \underset{g(x)}{\operatorname{argmin}} \sum_i \left(y_i - g\left(x_i\right)\right)^2$$

Identification and the curse of dimensionality

- In order to obtain a well defined problem, further assumptions on $g(x_i)$ must be added
  - constraints on eligible functions $g(x_i)$
  - constraints on the set of controls x_i
- Second option not practical in many situations:
  - If 100 observations represents a dense sample for a single input system, then for K inputs, 100^K
  - all observations are close to an "edge" of the sample

Penalty

- One way of overcoming these problems is by incorporating a penalty in the problem

$$\widehat{\mathrm{E}}\left(y|x\right) = \underset{g(x)}{\operatorname{argmin}} \sum_{i} \left\{ \left(y_i - g\left(x_i\right)\right)^2 + \lambda\phi\left(g\left(x_i\right)\right) \right\}$$

- The best fit is given by the solution without penalty, $\lambda = 0$
  - very low predictive power (overfitting)
- Common approach: divide the sample into a learning and a test sample

Examples of predictive learning

- least squares: $\phi(g(x)) = \begin{cases} \infty \text{ if } g(x) \neq h(x|\theta) \\ 0 \text{ otherwise} \end{cases}$
  - $h(\cdot)$ and θ are known
  - hence
$$\widehat{\mathrm{E}}(y|x) = \underset{g(x)}{\operatorname{argmin}} \sum_i \left\{ (y_i - h(x_i|\theta))^2 \right\}$$
- single layer neural network: $g(x) = \sum_t a_t \mathrm{s}(x'\theta_t)$
  - $\mathrm{s}(\cdot)$ is a sigmoid function
- projection pursuit: $g(x) = \sum_t g_t(x'\theta_t|a_t)$

Tree structures

$$\phi\left(g\left(x\right)\right) = \begin{cases} \infty \text{ if } g\left(x\right) \neq \sum_{t \in T} a_t \times \prod_{j=1}^{K} \mathbb{1}\left(l_j < x_j \leq u_j\right) \\ 0 \text{ otherwise} \end{cases}$$

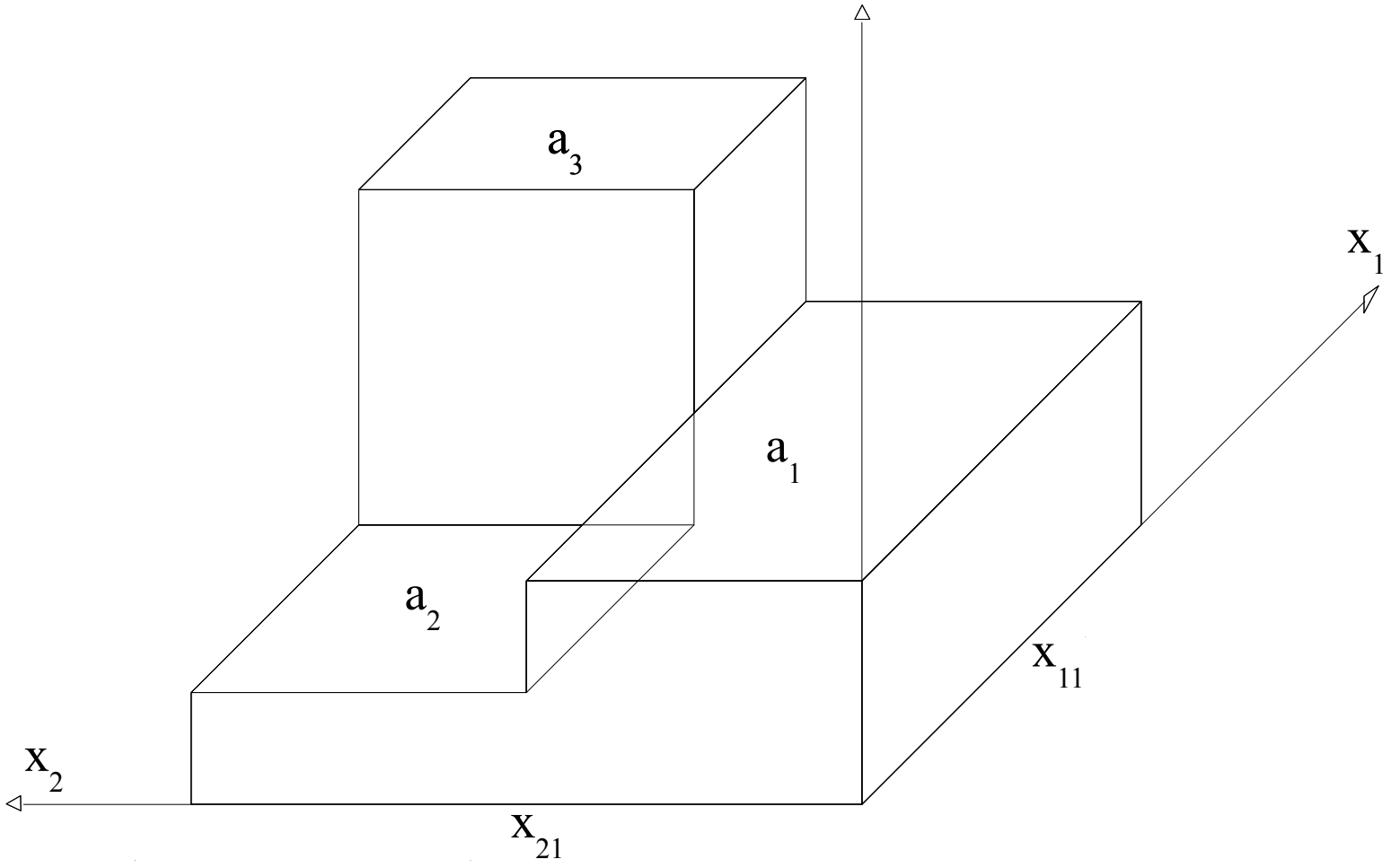
where l_j and u_j are the respective lower and upper limit of the region on each control

- T is a partition of the space of all possible values of x
- Therefore

$$\mathsf{E}\left(y|x\right) = a_t \times \prod_{j=1}^{K} \mathbb{1}\left(l_j < x_j \leq u_j\right)$$

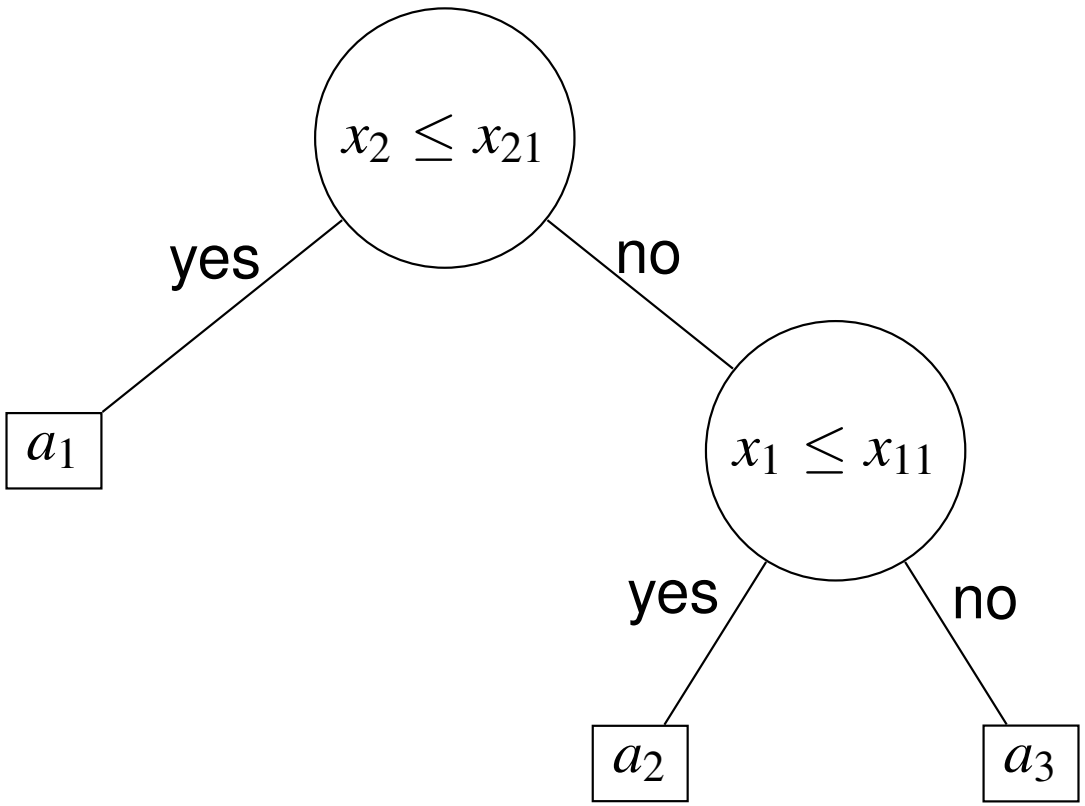
- Both the partition T and the expectations a_t associated to each element in the partition are unknown

Example



Mathematical and tree representation

$$\mathrm{E}\left(y|x_1,x_2\right) = \begin{cases} a_1 \text{ if } x_2 \leq x_{21} \\ a_2 \text{ if } x_2 > x_{21} \text{ and } x_1 \leq x_{11} \\ a_3 \text{ if } x_2 > x_{21} \text{ and } x_1 > x_{11} \end{cases}$$



Classification And Regression Trees

Estimation of tree structures

- The problem is if we know the tree structure: least squares
- Least squares is unfeasible when structure is unknown
  - LS on 50 cells with at most two terminal nodes $\approx 6 \times 10^{14}$ models (or more than 15 years of computing time)
- Second best solution: recursive partition
  - regions become more local
  - each step only considers a limited number of possible splits

Splitting algorithm in regression trees

- Assume that we have a tree structure T and that we want to split node t^*, one terminal node in T.
- Let $R(T)$ be the residual sum of squares within each terminal node of the tree.
- Consider the set of possible binary partitions or splits.

Recursive partitioning is defined by choosing the split at each step of the algorithm such that the reduction in $R(T)$ is maximized.

- The process ends with the largest possible tree, T_{MAX} where there are no nodes to split or the number of observations reach a lower limit (splitting rule).

Growing the tree until T_{MAX}

- Often, the result will be equivalent to dividing the sample into all possible cells and computing within-cell least squares.
- Growing the tree until no further partitioning is possible helps avoiding having to select a rule to stop splitting.
- Usually, however, T_{MAX} will be too complex in the sense that some terminal nodes could be aggregated into one terminal node.
- A more simplified structure will normally lead to more accurate estimates since the number of observations in each terminal node grows as aggregation takes place.
- It is also intuitive to see that if aggregation goes too far, aggregation bias will become a serious problem.

Pruning the tree: Error-complexity clustering

- In order to aggregate from T_{MAX} we can use a clustering algorithm procedure.
- For a given value α, let $R(\alpha, T) = R(T) + \alpha |T|$ where $|T|$ denotes the number of terminal nodes, or complexity, of the tree.
- The tree structured estimate for a given α, $T(\alpha)$, is the value that minimizes $R(\alpha, T)$ for the set of subtrees of T_{MAX}.
  - $T(\alpha)$ belongs to a much broader set than the sequence of all trees obtained in the recursive partition algorithm.
- For all α: $T_{MAX} \succ T(\alpha_1) \succ \ldots \succ \{\text{root}\}$ (pruning the tree)

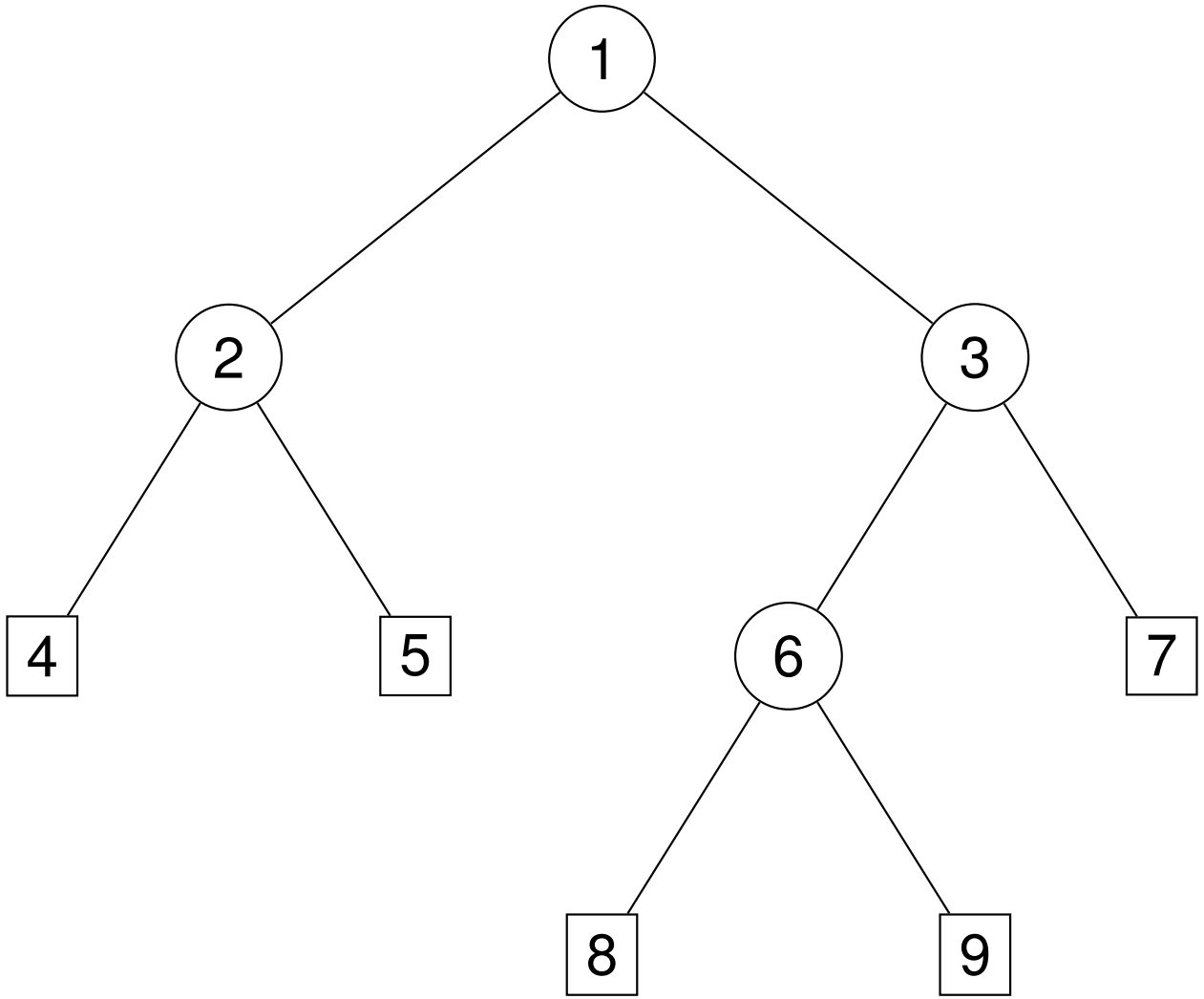
Honest tree

- By construction, $R\left(T_{MAX}\right)$ is the lowest value for the sequence of subtrees.
  - This may not be true for an independent sample: choosing T_{MAX} as our tree structured model may lead to overoptimistic results for $R\left(\cdot\right)$
- There are three strategies to obtain unbiased estimates of $R\left(\cdot\right)$:
  - test sample: choose the tree in the sequence that minimizes

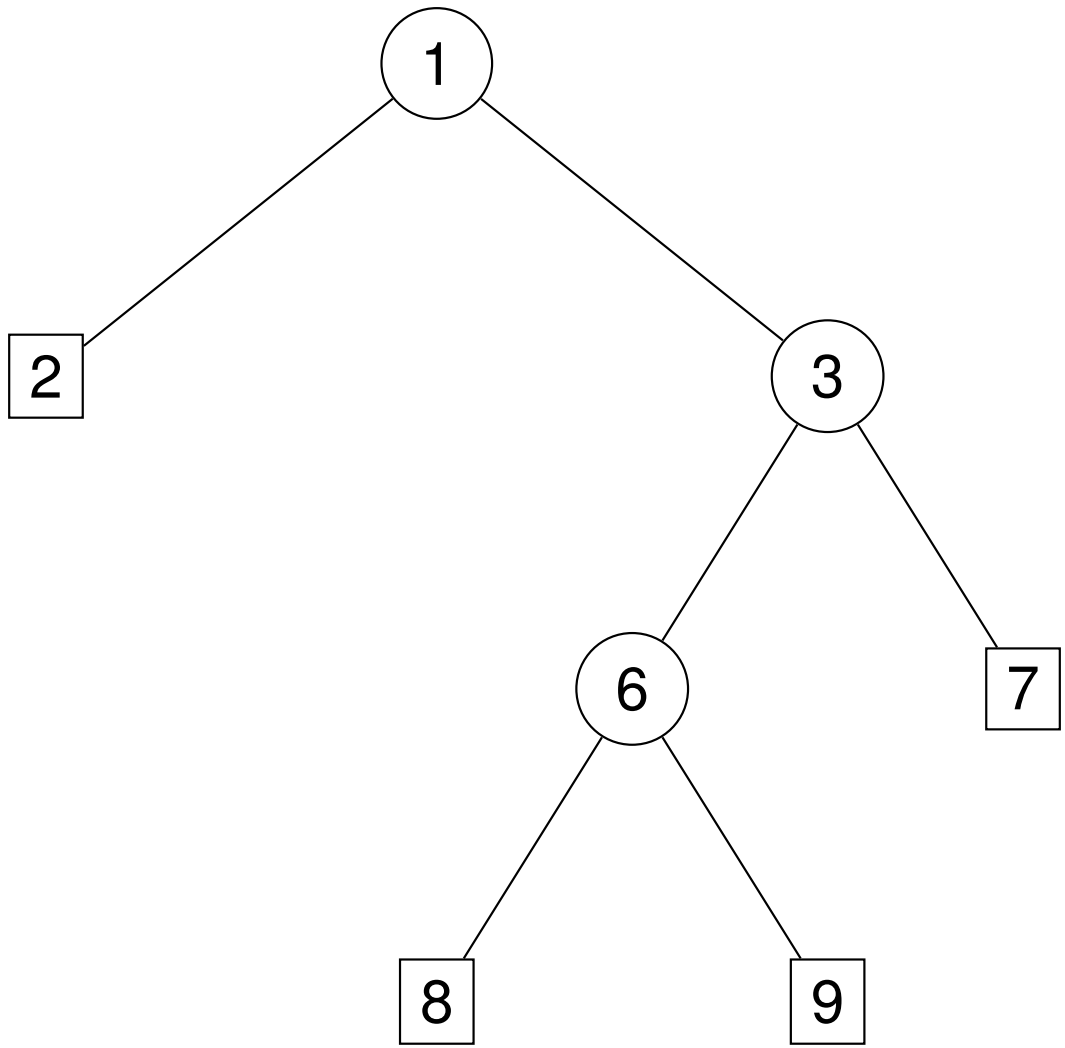
$$R^{ts}\left(T\right) + s \times \text{SE}\left(R^{ts}\left(T\right)\right)$$

    where s is a given positive value
  - K-fold cross validation
  - bootstrap

T_{MAX} example: 5 terminal nodes



T_1 example: 4 terminal nodes



T_2 example: 1 terminal node

- The sequence is thus: $\{T_{MAX}, T_1, T_2 \equiv \{\text{root}\}\}$
- Among the three, we would choose the tree that gives a smaller $R^{ts}(T) + s \times \text{SE}(R^{ts}(T))$
  - For example, $s = 1$ may be useful when the sequence provides a flat profile for $R^{ts}(T)$ after reaching a certain level of complexity

CART Estimator properties

- Consistency requires an ever more dense sample at all n-dimensional balls of the input space
- Cost-complexity minimization together with test sample unbiased estimates of $R(\cdot)$ guarantee that such condition is satisfied by regression tree partitions.
- The basic results can be found in Breiman et alia (1984, chapter 12).
- For small samples, high correlation in the explanatory variables will induce instability in the tree topology: interpretation of the contribution of each variable will become problematic

ARIES

The aries ado

aries *varname splitvarlist* [*if*] [*in*], options

- *varname*: output variable (it must be discrete if classification tree is performed)
- *splitvarlist*: variables whose combinations identify the terminal nodes
- By default, the command performs CART for regression trees with a constant in each terminal node using a test sample and the 0 SE rule for estimating the honest tree.

Options for regression trees

- **reg**ressors(*varlist*): controls in terminal nodes. A regression line is estimated in each terminal node.
- **exog**enous(*varlist*): list of exogenous variables. IV regression is estimated in each terminal node. The number of exogenous variables must be at least equal to the number of controls.
- noconstant: estimates regression lines without constant.

Options for classification trees

- Classification trees:
  - The output variable must be discrete.
  - Each value of the output variable refers to one of J classes.
  - Classification trees grows the tree using a given impurity measure based on the sample probability of each class in each node.
- Options for classification trees:
  - classification: performs classification tree (output variable must be discrete)
  - impurity(#): impurity measure code:
    - 1: Entropy measure
    - 2: Gini measure

Options common to classification and regression trees

- **seed**(#): seed to replicate random division of the sample into a learning and a test sample
- **lss**ize(#): proportion of the learning sample (default is 0.5)
- **stop**(#): integer for stop splitting rule
- rule(#): SE rule to identify honest tree

Output display

After regression trees:

- The overall fit of the model both for the learning and the test sample
- The definition of each terminal node in terms of the splitting variables
- The coefficient estimates and standard errors for each terminal node
  - The standard error of each terminal node regression is computed using the test sample

After classification trees:

- The overall miss-classification rate of the model estimated by test sample
- The definition of each terminal node in terms of the splitting variables
- The miss-classification rate for each terminal node in the learning and the test sample

Saved results

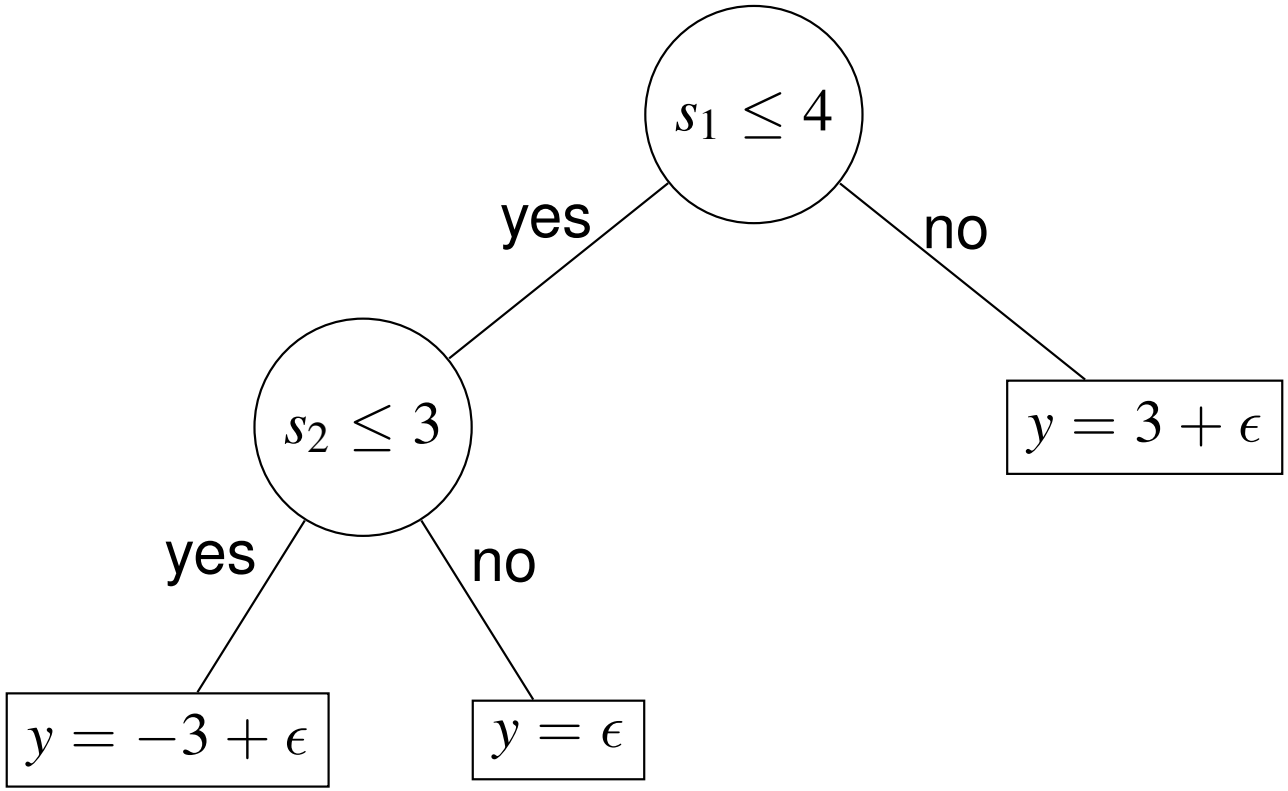
- Saved results for Regression trees:
  - usual scalars saved in e() after regresion
  - coefficients estimates and variance-covariance matrices for each terminal node's regression
- Common saved results:
  - a matrix representation of the tree structure
  - a matrix with range of values for splitting variables in each terminal node
  - a matrix with the sequence of optimal trees and the test-sample $R^{ts}(T)$ measure for each of them

Predictions

- `aries` saves the coefficients estimates and also matrix representations of the estimated tree
- `predict` is available after estimation
- After regression trees: `predict` *`newvar`* `[if] [in], xb residual nodes`
  - `xb`: output variable predictions (the default)
  - `residual`: residuals
  - `nodes`: terminal node code
- After classification trees: `predict` *`newvar`* `[if] [in]`
  - the variable *newvar* includes the class code predicted by the estimated tree for each observation

Simulations

Example 1: RT with constant



$$\epsilon \sim \mathcal{N}(0, 1)$$
$$s_1 \in \{2, 4, 6, 8\}, s_2 \in \{3, 6, 9, 12\}$$

aries y s1 s2, stop(5)

```
Learning Sample                                 Test Sample
Number of obs = 522                             Number of obs =     478
F(  2,   519) = 1135.5                          F(  2,   475) =  1066.1
Prob > F      = 0.0000                          Prob > F      =  0.0000
R-squared     = 0.8140                          R-squared     =  0.8178
Adj R-squared = 0.8133                          Adj R-squared =  0.8170
Root MSE      = 1.0217                          Root MSE      =  1.0077
```

Node 3: 6<=s1<=8 3<=s2<=12

No of obs (Learning smpl) = 257 — No of obs (Test smpl) = 239

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_cons	3.109557	.0622225	49.97	0.000	2.987603	3.23151

Node 4: 2<=s1<=4 3<=s2<=3

No of obs (Learning smpl) = 70 — No of obs (Test smpl) = 63

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_cons	-2.852275	.1054995	-27.04	0.000	-3.059051	-2.6455

Node 5: 2<=s1<=4 6<=s2<=12

No of obs (Learning smpl) = 195 — No of obs (Test smpl) = 176

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_cons	-.0097753	.0760195	-0.13	0.898	-.1587707	.1392202

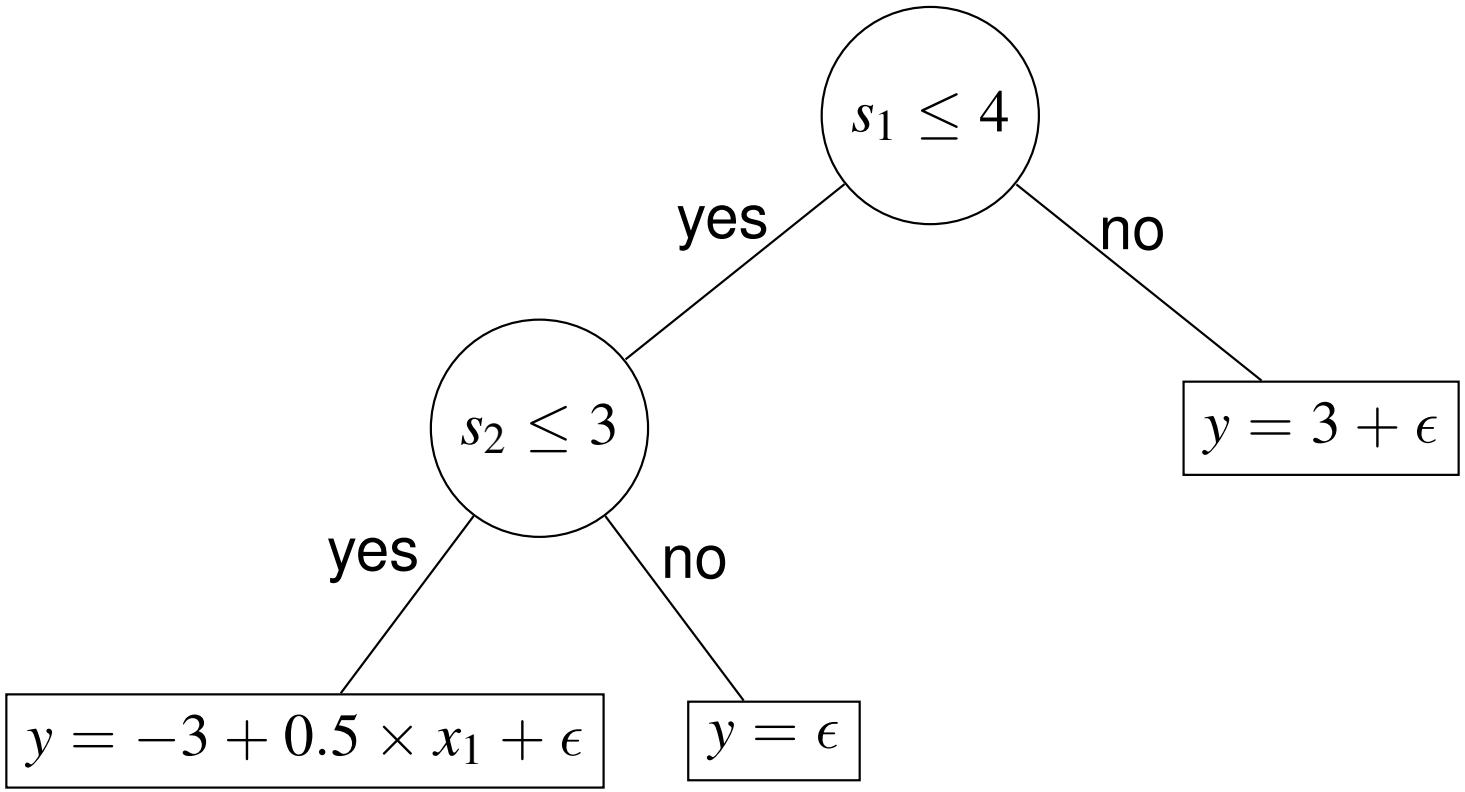
A simple Monte Carlo

Table: Monte Carlo: R^2

No. obs.	σ	OLS	aries:LS	aries:TS
250	.5	0.711	0.946	0.946
250	1	0.612	0.814	0.815
250	2	0.396	0.525	0.528
750	.5	0.710	0.946	0.947
750	1	0.611	0.813	0.816
750	2	0.393	0.520	0.529
1000	.5	0.711	0.946	0.946
1000	1	0.612	0.814	0.815
1000	2	0.393	0.523	0.524

Note: Monte Carlo results using 500 replications.

Example 2: RT with regression line



$$\epsilon \sim \mathcal{N}(0, 1)$$
$$s_1 \in \{2, 4, 6, 8\}, s_2 \in \{3, 6, 9, 12\}$$

aries y s1 s2, reg(x1) stop(5)

```
Learning Sample                          Test Sample
Number of obs = 522                      Number of obs =     478
F(  5,   516) = 339.95                   F(  5,   472) =  339.11
Prob > F      = 0.0000                   Prob > F      =  0.0000
R-squared     = 0.7671                   R-squared     =  0.7822
Adj R-squared = 0.7649                   Adj R-squared =  0.7799
Root MSE      = 1.0113                   Root MSE      =  0.9746
```

Node 3: 6<=s1<=8 3<=s2<=12

No of obs (Learning smpl) = 257 — No of obs (Test smpl) = 239

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	-.0738962	.0543166	-1.36	0.174	-.1803548	.0325624
_cons	3.229408	.1481916	21.79	0.000	2.938958	3.519859

Node 4: 2<=s1<=4 3<=s2<=3

No of obs (Learning smpl) = 70 — No of obs (Test smpl) = 63

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	.5736499	.1053622	5.44	0.000	.3671437	.780156
_cons	-3.173849	.2846731	-11.15	0.000	-3.731798	-2.6159

Node 5: 2<=s1<=4 6<=s2<=12

No of obs (Learning smpl) = 195 — No of obs (Test smpl) = 176

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	.0250965	.0669443	0.37	0.708	-.106112	.1563049
_cons	.054355	.1829895	0.30	0.766	-.3042978	.4130078

Some saved results

```
matrix list e(tree)
```

```
e(tree)[5,4]
          Node      Child  Split_var    Cut_off
r1           1          2          1          5
r2           2          4          2        4.5
r3           3          0          0          0
r4           4          0          0          0
r5           5          0          0          0
```

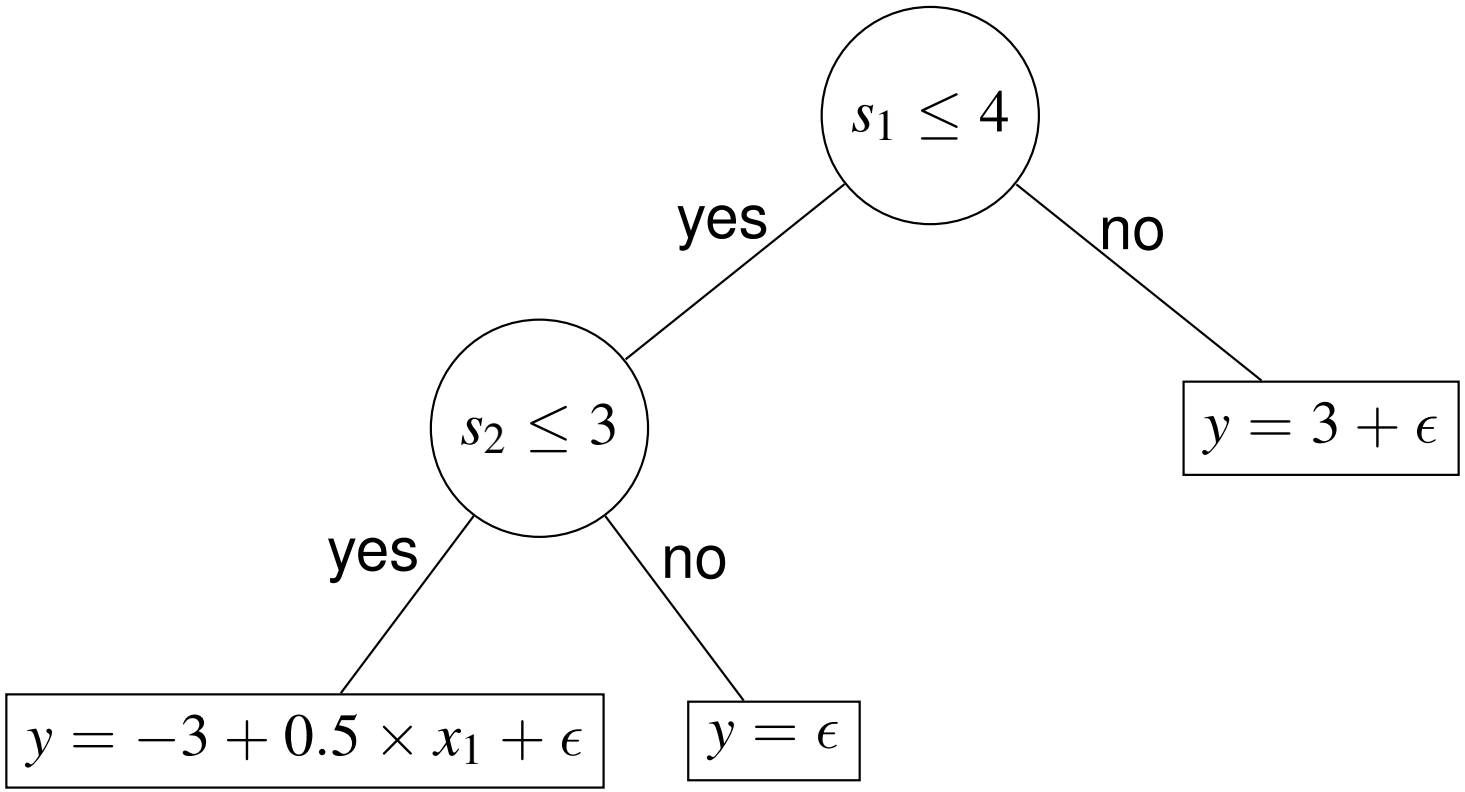
```
matrix list e(_tree)
```

```
e(_tree)[5,5]
       Node  s1_min  s1_max  s2_min  s2_max
r1        1       2       8       3      12
r2        2       2       4       3      12
r3        3       6       8       3      12
r4        4       2       4       3       3
r5        5       2       4       6      12
```

```
matrix list e(pruning)
```

```
e(pruning)[12,2]
     Complexity    Impurity
 r1           1   4.2953238
 r2           2   1.3388132
 r3           3   .94715324
 r4           4   .94849036
 r5           6   .95739838
 r6          10   .98367937
 r7          11   .99016994
 r8          12   .98654955
 r9          13    .9930246
r10          14   .99768252
r11          15   .99777621
r12          16   .99712844
```

Example 3: RT with IV line

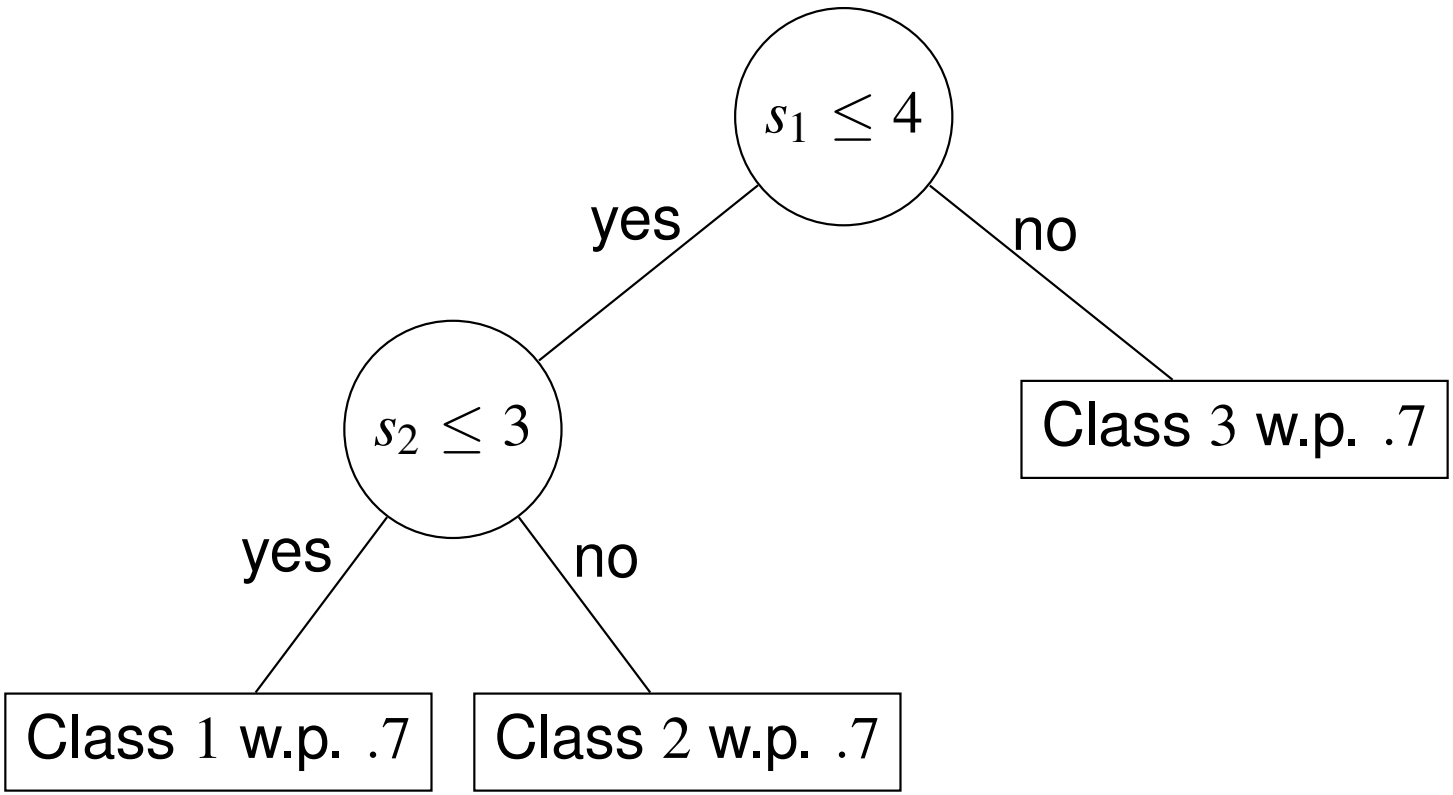


$$\epsilon \sim \mathcal{N}(0,1),\ \text{cov}(x_1, \epsilon) \neq 0,\ \text{cov}(z_1, \epsilon) = 0$$
$$s_1 \in \{2,4,6,8\}, s_2 \in \{3,6,9,12\}$$

aries y s1 s2, reg(x1) exog(z1) stop(5)

```
Learning Sample                                  Test Sample
Number of obs = 522                              Number of obs =     478
F(  5,   516) = 386.75                           F(  5,   472) =  392.25
Prob > F      = 0.0000                           Prob > F      =  0.0000
R-squared     = 0.7899                           R-squared     =  0.8066
Adj R-squared = 0.7879                           Adj R-squared =  0.8046
Root MSE      = 1.0190                           Root MSE      =  0.9797

Node 3:  6<=s1<=8 3<=s2<=12
No of obs (Learning smpl) = 257                  No of obs (Test smpl) =     239
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   -.152294   .1119791    -1.36   0.174     -.371769    .0671811
       _cons |   3.236396   .1529517    21.16   0.000     2.936616    3.536176
------------------------------------------------------------------------------

Exogenous variable: z1

Node 4:  2<=s1<=4 3<=s2<=3
No of obs (Learning smpl) = 70                   No of obs (Test smpl) =      63
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .6430845   .2088366     3.08   0.002     .2337722    1.052397
       _cons |  -3.168874   .2838073   -11.17   0.000    -3.725126   -2.612622
------------------------------------------------------------------------------

Exogenous variable: z1

Node 5:  2<=s1<=4 6<=s2<=12
No of obs (Learning smpl) = 195                  No of obs (Test smpl) =     176
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .0496941   .1334359     0.37   0.710    -.2118355    .3112237
       _cons |   .0538148   .1855386     0.29   0.772    -.3098342    .4174638
------------------------------------------------------------------------------

Exogenous variable: z1
```

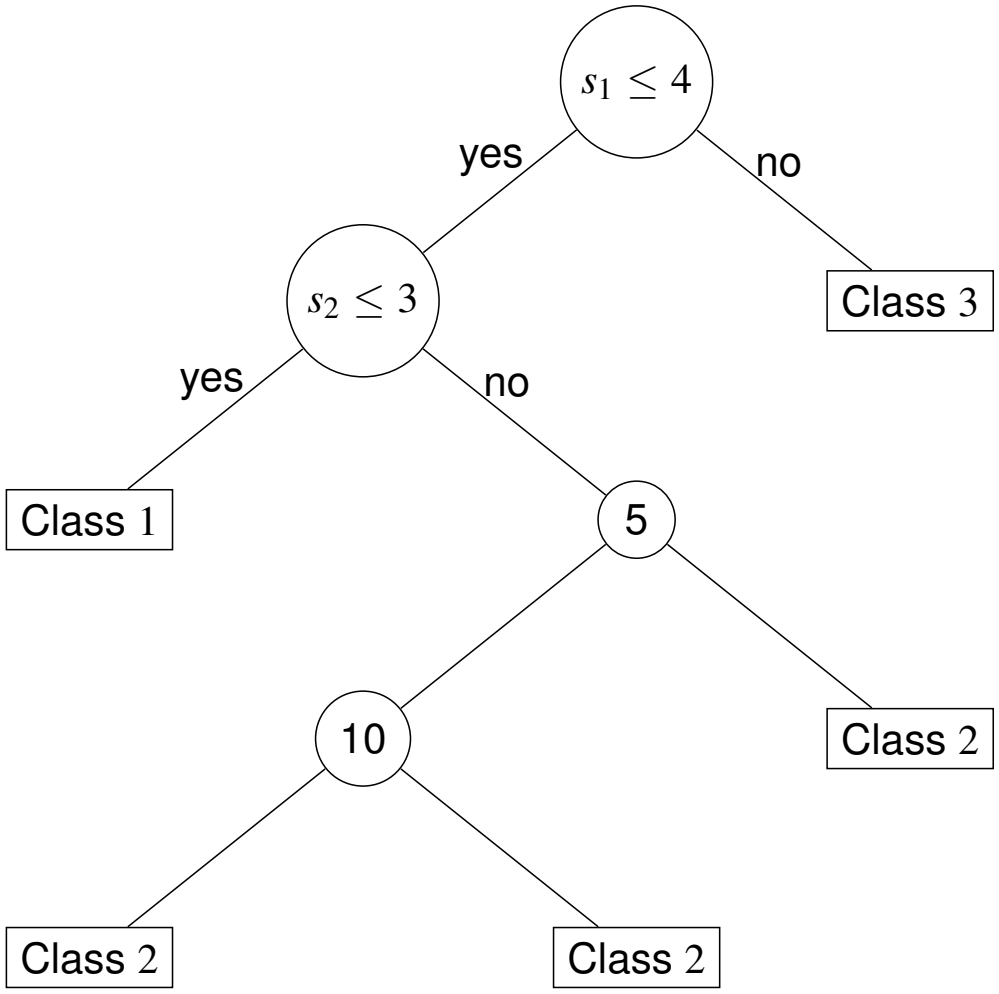
Example 4: Classification trees



$$s_1 \in \{2, 4, 6, 8\}, s_2 \in \{3, 6, 9, 12\}$$

aries y s1 s2, class

```
Learning sample (no.obs):    522
Test sample (no.obs):        478
No. of terminal nodels:      5
Pr. of missclassification:   0.3096

Node 3:  6<=s1<=8 3<=s2<=12
Class:3                      Learning Sample          Test Sample
Pr(missclassification)       0.2918                   0.2762
No. of obs.                  257                      239

Node 4:  2<=s1<=4 3<=s2<=3
Class:1                      Learning Sample          Test Sample
Pr(missclassification)       0.3000                   0.2857
No. of obs.                  70                       63

Node 11:  2<=s1<=4 12<=s2<=12
Class:2                      Learning Sample          Test Sample
Pr(missclassification)       0.2373                   0.3729
No. of obs.                  59                       59

Node 16:  2<=s1<=2 6<=s2<=9
Class:2                      Learning Sample          Test Sample
Pr(missclassification)       0.2329                   0.3770
No. of obs.                  73                       61

Node 17:  4<=s1<=4 6<=s2<=9
Class:2                      Learning Sample          Test Sample
Pr(missclassification)       0.3333                   0.3393
No. of obs.                  63                       56
```


$s_1 \leq 4$
yes
no
$s_2 \leq 3$
Class 3
yes
no
Class 1
5
10
Class 2
Class 2
Class 2

Extensions

- v-fold cross-validation for small data sets
- combining splitting variables in a single step
- categorical splitting variables
- graphs producing tree representation and sequence of $R^{ts}(T)$ estimates
- alternative impurity measurements
- boosting

Thank you