Using and Improving Decision Trees in R

Orange County R User Group

Robert Mohr

January 23, 2018

Packages and References

There are a couple of standard decision tree packages, used by a couple of prominent modeling books:

- **tree**, *An Introduction to Statistical Learning* by James, Witten, Hastie, and Tibshirani
- **rpart**, *Applied Predictive Modeling* by Kuhn and Johnson

The visualizations supported by these packages are based on R's base graphics system. We are developing a script, gridTree.R, that leverages the more powerful graphics system of the **grid** package.

Today, we will focus on **tree** and gridTree.R.

tree: Fitting the data

Fitting is done with one call to tree(). tree() uses standard formula notation:

```r
library(tree)

tree.iris <- tree(Species~., data=iris)
```

This works for both classification and regression trees.

Exploring the tree object: raw

```
tree.iris
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
##    2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.
##    3) Petal.Length > 2.45 100 138.600 versicolor ( 0.000
##      6) Petal.Width < 1.75 54  33.320 versicolor ( 0.000
##       12) Petal.Length < 4.95 48   9.721 versicolor ( 0.
##         24) Sepal.Length < 5.15 5   5.004 versicolor ( 0
##         25) Sepal.Length > 5.15 43   0.000 versicolor (
##       13) Petal.Length > 4.95 6   7.638 virginica ( 0.00
##      7) Petal.Width > 1.75 46   9.635 virginica ( 0.0000
##       14) Petal.Length < 4.95 6   5.407 virginica ( 0.00
##       15) Petal.Length > 4.95 40   0.000 virginica ( 0.0
```

Exploring the tree object: str()

```
str(tree.iris, give.attr = F)
```

```
## List of 6
##  $ frame  :'data.frame': 11 obs. of  6 variables:
##   ..$ var    : Factor w/ 5 levels "<leaf>","Sepal.Length'
##   ..$ n      : num [1:11] 150 50 100 54 48 5 43 6 46 6 ..
##   ..$ dev    : num [1:11] 329.58 0 138.63 33.32 9.72 ...
##   ..$ yval   : Factor w/ 3 levels "setosa","versicolor",
##   ..$ splits: chr [1:11, 1:2] "<2.45" "" "<1.75" "<4.95'
##   ..$ yprob : num [1:11, 1:3] 0.333 1 0 0 0 ...
##  $ where  : Named int [1:150] 2 2 2 2 2 2 2 2 2 2 ...
##  $ terms  :Classes 'terms', 'formula'  language Species
##  $ call   : language tree(formula = Species ~ ., data =
##  $ y      : Factor w/ 3 levels "setosa","versicolor",..
##  $ weights: num [1:150] 1 1 1 1 1 1 1 1 1 1 ...
```

Exploring the tree object: $frame

```r
str(tree.iris$frame)
```

```
## 'data.frame':    11 obs. of  6 variables:
##  $ var   : Factor w/ 5 levels "<leaf>","Sepal.Length",..
##  $ n     : num  150 50 100 54 48 5 43 6 46 6 ...
##  $ dev   : num  329.58 0 138.63 33.32 9.72 ...
##  $ yval  : Factor w/ 3 levels "setosa","versicolor",..:
##  $ splits: chr [1:11, 1:2] "<2.45" "" "<1.75" "<4.95" ..
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr  "cutleft" "cutright"
##  $ yprob : num [1:11, 1:3] 0.333 1 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr  "setosa" "versicolor" "virginica"
```

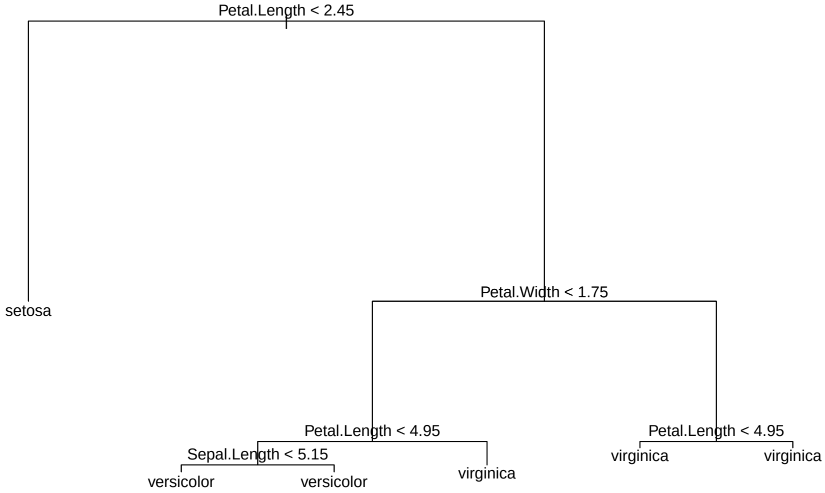
tree: Plotting the results

Plotting is straightforward, thanks to methods that **tree** provides for plot() and text() for objects of class tree.

```
# Calls to inspect the tree methods
# methods(plot)
# getS3method("plot", "tree")
# methods(text)
# getS3method("text", "tree")

# Two calls to render everything (results on next slide)
# plot(tree.iris)
# text(tree.iris)
```

A **tree** plot of iris



Opportunities for improvement: orientation

Decision trees are traditionally drawn top-down. This has a couple of disadvantages:

- The major axis of text is horizontal rather than vertical, so overlap is inevitable.
- Labeling the branching criteria ($<$ x, $>$ y) usually involves dropping one label, which is ambiguous, or spreading labels out, which is inefficient.

Creating a left-right oriented decision tree would be generally easier to read.

Opportunities for improvement: color encoding

Traditional decision trees make use of almost no data-encoding visual attributes.

For example, nodes could be color-encoded to represent:

-Classifications, as a redundancy to direct labelling, or in place of it
-Diagnostic measures, such as node purity. Keep in mind this is effective for rough distinctions only.

A very brief introduction to **grid**

- **grid** is a powerful graphics package that underlies **ggplot2** and **lattice** graphics
- **grid** can be used to modify these graphics or create entirely new ones
- Part of **grid**'s power comes from trees of *viewport* objects

For more detail, see *R Graphics* by Paul Murrell (the creator of **grid**).

A very brief introduction to viewports

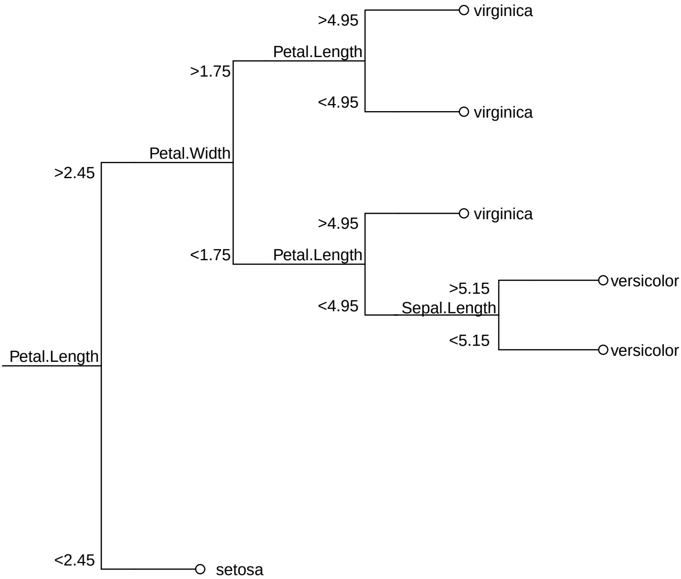
- A viewport is a rectangular drawing region
- Viewports have a *geometric context* (coordinate systems, size) and a *graphical context* (graphical parameters like fontsize and colors)
- Viewports have a special coordinate system known as *normalized parent coordinates* (npc) with (0,0) at the lower left corner and (1,1) in the upper right corner
- Viewports can have children that inherit their properties

Plotting with gridTree

gridTree.R takes advantage of the fact that both decision trees and viewports have a tree structure:

- grid.tree() accepts an object of class tree, extracts the data frame, and calls grid.grow() for the first tree node
- grid.grow() draws a part of the tree by calling grid.branch() for a split or grid.leaf() for a terminal node
- grid.grow() recursively calls itself for the children of the current node
- The process naturally terminates when there are no more nodes

A gridTree rendering of iris



Desirable features of the gridTree

- Can be easily read left-to-right
- No overlap of text with the tree
- Variables and their bounds are closely grouped and aligned
- Terminal nodes can be color-coded with supporting data

Future work on gridTree

- Implementation of colored nodes
- Checks and evasions for colliding branches
- Checks and evasions for colliding text
- Implementation of graphical objects (grobs)

Repository on github:
`https://github.com/mohrsignal/gridTree`

Email `mohrsignal@gmail.com` with questions.

Thanks for your time and attention!