Let's Have a party! An Open-Source Toolbox for Recursive Partytioning

Torsten Hothorn, Achim Zeileis, Kurt Hornik

Department of Statistics and Mathematics
Wirtschaftsuniversität Wien

Research Report Series

Report 59
September 2007

http://statmath.wu-wien.ac.at/

Let's Have a *party*! An Open-Source Toolbox for Recursive Partytioning

Torsten Hothorn and **Achim Zeileis** and **Kurt Hornik**
R Project for Statistical Computing and Graphics
E-mail: Firstname.Lastname@R-project.org

Abstract

Package *party*, implemented in the R system for statistical computing, provides basic classes and methods for recursive partitioning along with reference implementations for three recently-suggested tree-based learners: conditional inference trees and forests, and model-based recursive partitioning.

Keywords: open-source software, recursive partitioning, decision trees, statistical learning.

1. Overview

Tree-based learners, also known as recursive partitioning techniques, belong to the core tools for classification and regression, both in the machine learning and statistical learning communities. Among the most popular representatives are C4.5 (Quinlan, 1993) and CART (Breiman et al., 1984), respectively, which were both accompanied by non-free implementations of the algorithms. Following the success of these seminal algorithms, many modifications, extensions, and enhancements of the basic recursive partitioning idea have been proposed in the literature—more often than not with proprietary or closed-source implementations or no software at all. A notable exception is the open-source Java package *Weka* (Witten and Frank, 2005) that provides free de facto reference implementations of tree-based learners (including C4.5) or original implementations of novel algorithms (such as LMT). In statistical learning, the "lingua franca" is R (R Development Core Team, 2007), an open-source programming environment that also provides a reference implementation for CART in package *rpart*.

Here, we present the R package *party*, a toolbox for recursive partytioning. Unlike the implementations above, it does not only implement specific algorithms, but aims to provide basic infrastructure for tree-based learners along with flexible and adjustable learning algorithms built on top. The basic data structures encompass a general high-level class `BinaryTree` defined in R for representing various types of binary trees where arbitrary types of information can be attached to the nodes. It comes with generic functionality for summarizing such trees, creating visualizations and computing predictions. This generic infrastructure has been employed for implementing three flexible recursive partitioning algorithms. The first is the conditional inference tree framework of Hothorn et al. (2006) which learns classical trees with constant fits in the terminal nodes, based on permutation tests for split selection. Its user interface is the function `ctree` that can be applied out of the box to regression and classification problems where both inputs and outputs can be categorical, numerical, censored, or multivariate. Re-using this basic learner, the package also provides an ensemble method for fitting forests of trees in the function `cforest`. A key advantage of this random forest type algorithm is that it can produce unbiased variable importance measures (Strobl et al., 2007). Finally, function `mob` provides a model-based recursive partitioning learner (Zeileis et al., 2008) that allows to attach arbitrary parametric models (such as, e.g., linear or logistic regression, multivariate or survival models) to the nodes in the tree and uses structural change tests for split selection.

The package *party* and all the components it depends on—including packages with tools for model construction, with visualization infrastructure and with inference techniques—are available under the General Public License (GPL) from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/. The package contains R and C code and has been successfully built on a multitude of platforms including Windows, Mac OS and various flavours of Linux. It can be installed automatically from within R via the command `install.packages("party", dependencies = TRUE)` (which also installs all packages needed for examples); to get started, see the online reference manual, e.g., via `help(package = "party")`. More advanced user- and developer-level documentation is available in the package vignettes (see `vignette(package = "party")`) and the C source code *doxygen* documentation (in the `inst/documentation` sub-directory of the package sources).

In the following, we give a brief tour of the package by applying the three core algorithms to classification problems before discussing some details of the developer-level features.

2. User interface

To demonstrate usage of the high-level algorithms, we present some short examples for fitting single trees, random forests and model-based trees in *party* for two different binary classification tasks.

Trees and forests. The `GlaucomaM` data set contains 196 observations of glaucoma classification in human eyes (affected vs. healthy) along with 62 features derived from confocal laser scanning images of the optic nerve head. In the R session below, we first load package and data and then split it randomly into training and test sets of 150 and 46 observations, respectively.

```
R> library("party")
R> data("GlaucomaM", package = "ipred")
R> set.seed(10131218)
R> idx <- sample(1:nrow(GlaucomaM), 150)
R> GM1 <- GlaucomaM[idx,]
R> GM2 <- GlaucomaM[-idx,]
```

A `ctree` for predicting the `Class` variable, as explained by (`~`) all other variables (`.`) contained in the learning set `GM1`, can be fitted via

```
R> ctGM <- ctree(Class ~ ., data = GM1)
```

creating a fitted `BinaryTree` object `ctGM`. For obtaining a short summary of the tree it can be either printed by typing `ctGM` at the command line (or more explicitly: `print(ctGM)`) or visualized by `plot(ctGM)`. The resulting plot (depicted in the left panel of Figure 1) shows that the glaucoma probability increases clearly for lower eye volumes (as captured in the different volume measures `vari` and `vasg`). After three splits, no more significant splits were found. To evaluate the performance of this fitted tree on the test data `GM2`, we can compute the confusion matrix of true and predicted class memberships

```
R> table(true = GM2$Class, pred = predict(ctGM, newdata = GM2))
```

```
          pred
true        glaucoma normal
  glaucoma        15      7
  normal           2     22
```

corresponding to a misclassification rate of 19.6%. Of course, this is only a rough ad hoc estimate of the generalization error, for a more thorough evaluation and comparison with other tree algorithms see the original publications mentioned above.

Similarly, we can fit a whole random forest of `ctree`s and compute their confusion matrix

```
R> cfGM <- cforest(Class ~ ., data = GM1)
R> table(true = GM2$Class, pred = predict(cfGM, newdata = GM2))
```

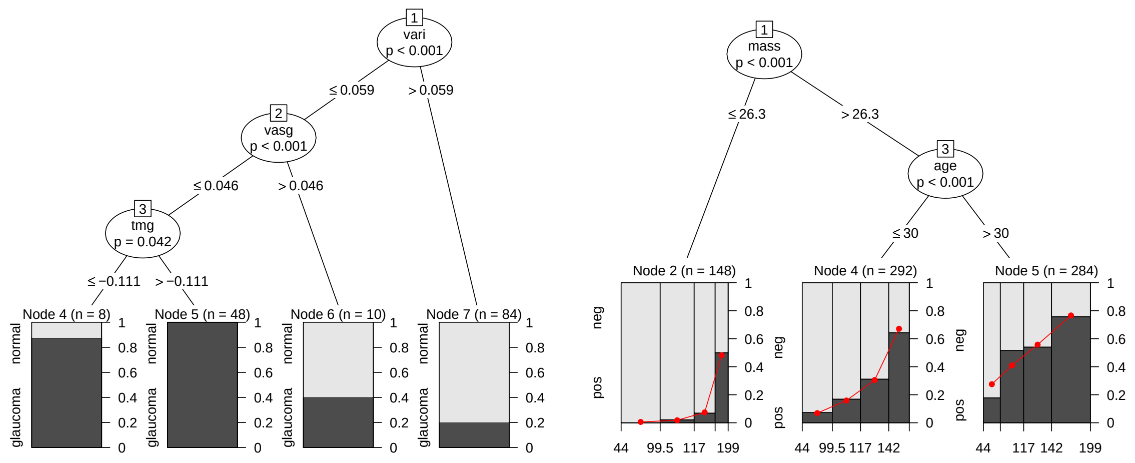

Figure 1: Tree visualizations: distribution of glaucoma for fitted `ctree` (left) and distribution of diabetes over glucose for logistic-regression `mob` (right).

```
          pred
true       glaucoma normal
  glaucoma       19      3
  normal          3     21
```

corresponding to a misclassification rate of 13.0% Not surprisingly, `cforest` improves on `ctree`, however the forest does not have an intuitive visualization like the single tree (see Figure 1).

Model trees. For illustrating the `mob` algorithm, we partition a logistic regression for `diabetes` explained by (`~`) `glucose` in the well-known Pima Indians Diabetes data set, using five remaining variables for partitioning.

```
R> data("PimaIndiansDiabetes", package = "mlbench")
R> PID <- subset(PimaIndiansDiabetes, glucose > 0 & pressure > 0 & mass > 0)
```

This loads the data set and selects a subset `PID` of 724 observations excluding physically impossible zero measurements in some of the variables. The `mob` based on a logistic regression (i.e., a generalized linear model with binomial family) can be fitted via

```
R> mobPID <- mob(diabetes ~ glucose | pregnant + pressure + mass + pedigree + age,
+    data = PID, model = glinearModel, family = binomial())
```

Its visualization produced by `plot(mobPID)` is depicted in the right panel of Figure 1. It shows the empirical `diabetes` proportions over `glucose` groups along with the predictions from the logisitc regression, conveying that the overall level (regression intercept) increases with body mass index and age while the glucose effect (regression slope) decreases. For a benchmark comparison of this algorithm with other trees (including *Weka*'s LMT and J4.8 implementation of C4.5) see Zeileis et al. (2008).

3. Internal structure

The *party* package contains R code defining classes and methods along with high-level algorithms built on top of these, as well as an additional program layer in C that codes time-critical operations directly modifying previously created R objects. In particular, the R building blocks encompass classes and methods for `LearningSample` and `BinaryTree` objects which can be re-used in high-level algorithms (such as `ctree`, `cforest` and `mob`) or directly at C level. `LearningSample` is a data set—created from external sources or conversion from other objects—which is enhanced by meta-information such as variable orderings that can speed up split searches. `BinaryTree` is a tree representation based on a recursive list structure containing node information (e.g., primary splits, surrogate splits, summary statistics, predictions, etc.) as well as left and right daughter nodes. Recursive methods for predicting future cases, textual and graphical representations etc., are defined for objects of class `BinaryTree`.

The tree-growing can be controlled in various ways: starting from simple hyper-parameters, such as minimal leaf size, significance level or maximal depth, to very complex arguments (e.g., parametric model classes for `mob` or visualization functions for terminal nodes). At the R level, users and developers can make use of the documented API for inspecting and manipulating `BinaryTree` objects and, in addition, are free to extend currently implemented methodology.

Unit testing of *party* (as for more than 1,200 other R packages on CRAN) is performed automatically on a daily basis for various platforms and versions of R.

4. Conclusions

The package *party* extends the set of machine learning techniques available in the R system for statistical computing (see the corresponding CRAN task view at http://CRAN.R-project.org/src/contrib/Views/) with a toolbox for recursive partitioning. The package provides basic infrastructure for binary trees along with reference implementations of three recently-suggested tree-based learners: conditional inference trees and forests, and model-based recursive partitioning. Party on!

References

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, California, 1984.

Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006. doi: 10.1198/106186006X133933.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. URL http://www.R-project.org/. ISBN 3-900051-07-0.

Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(25), 2007. doi: 10.1186/1471-2105-8-25. URL http://www.BioMedCentral.com/1471-2105/8/25/.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Achim Zeileis, Torsten Hothorn, and Kurt Hornik. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 2008. Forthcoming.