An Introduction to Recursive Partitioning for Heterogeneous Causal Effects Estimation Using `causalTree` package

Susan Athey
Guido Imbens
Yanyang Kong

April 13, 2016

Contents

1 Introduction

This document is a brief introduction of `causalTree` package, which is indended to give a short overview of rules and methods in `causalTree` function, as complementary material for Athey and Imbens's paper *Recursive Partitioning for Heterogeneous Causal Effects* [1].

The `causalTree` function builds a regression model using the `rpart` object, which is the object derived from `rpart` package, implemneting many ideas in the CART (Classification and Regression Trees), written by Breiman, Friedman, Olshen and Stone [2]. Like `rpart`, `causalTree` builds a binary regression tree model in two stages, but focuses on estimating heterogeneous causal effect.

As same as `rpart`, in the first stage, the tree is growing from root node based on specific splitting rule. In each node, data will be splitted into two groups to best minimize the risk function, then in left sub-node and right sub-node, the splitting routine will be applied seperately and so on recursively till no improvements can made or reach some limiting rules (eg. at least `minsize` of treated observations and controlled observations in terminal nodes.)

In the second stage, the tree will be pruned using some cross validation method to trim off some branches from the full tree in order to minize the cross validation error.

Something differs from `rpart`, in `causalTree` package, we offer honest re-estimation `honest.causalTree` on causal effects. Honest here means that we estimate causal effects on observed samples itself rather than the data used to build model and do cross validation.

2 Notations

X_i	$i = 1, 2, ..., n$	observed variables or feature matrix for observation i.
Y_i	$i = 1, 2, ..., n$	observed outcome of observation i.
W_i	$i = 1, 2, ..., n$	binary indicator for the treatment, with $W_i = 0$ indicating that observation i received the control treatment, and $W_i = 1$ indicating that observation i received the active treatment.
Π		partitioning tree as $\Pi = \bigcup L_l$, with L_l represents the $l-$th leaf.
$\tau(l)$	$l = 1, 2, ..., k$	causal effect or treatment effect in $l-$th leaf.
p		propensity score to indicate the treatment probability.

3 Building Causal Trees

3.1 Splitting rules

`causalTree` function offers four different splitting rules for user to choose. Each splitting rule corresponds to a specific risk function, and each split at a node aims to minimize the risk function.

For each observation i, its treatment effect or causal effect is estimated as the difference of treated mean and controlled mean in the leaf l where it belongs.

$$\hat{\tau}(i) = \hat{\tau}(l) = \frac{\sum_{W_i=1} Y_i}{n_1} - \frac{\sum_{W_i=0} Y_i}{n_0}$$

3.1.1 TOT

We first define the transformed outcome as

$$Y_i^* = Y_i \cdot \frac{W_i - e(X_i)}{e(X_i) \cdot (1 - e(X_i))}$$

where $e(x) = P(W_i = 1 | X_i = x)$ is the conditional treatment probability. In complete randomization, we can set $e(x) = p$ and the transformed outcome becomes

$$Y_i^* = \begin{cases} Y_i/p & W_i = 1 \\ -Y_i/(1-p) & W_i = 0 \end{cases}$$

In TOT splitting rule, the risk function is given by

$$E(\Pi; Y, W, X) = \frac{1}{n} \sum (Y_i^* - \hat{\tau}_i)^2$$

3.1.2 CT

In causal trees splitting rule, we have two versions, adaptive verison, denoted as CT-A, and honest version, CT-H.

3.1.3 fit

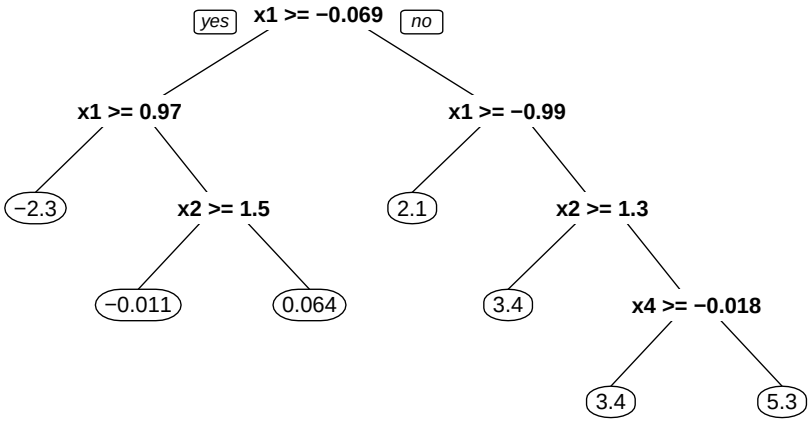
3.1.4 tstats

3.2 Discrete splitting

3.3 Example

The data we use in this example is the simulated data called `simulation.1` built in `causalTree` package.

```
> library(causalTree)
> fit <- causalTree(y~x1 + x2 + x3 + x4, data = simulation.1,
+                   treatment = simulation.1$treatment, split.Rule = "TOT",
+                   cv.option = "fit", cv.Honest = F, split.Bucket = F,
+                   xval = 10, cv.alpha = 0.5, propensity = 0.5)
> rpart.plot(fit)
```



4 Cross Validation and Pruning

4.1 Cross validation options

4.1.1 TOT

4.1.2 CT

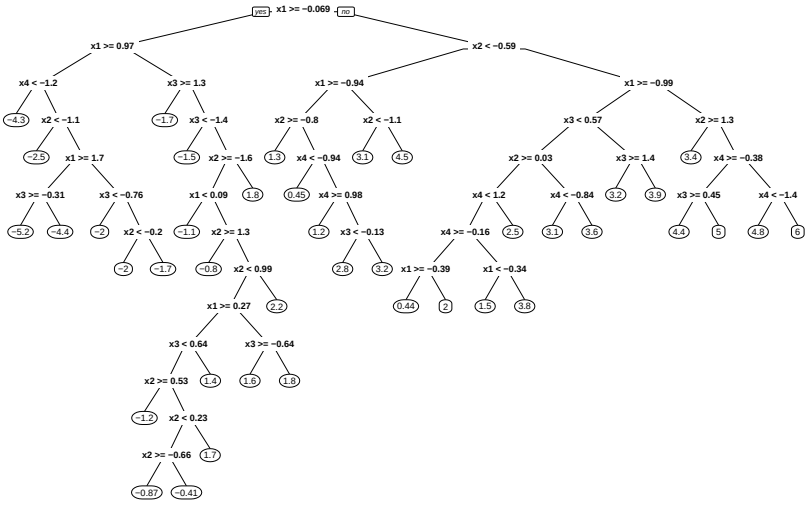
4.1.3 fit

4.1.4 matching

4.2 Example

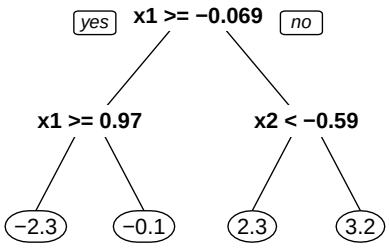
```
> fit <- causalTree(y~x1 + x2 + x3 + x4, data = simulation.1,
+                   treatment = simulation.1$treatment, split.Rule = "CT",
+                   cv.option = "CT", cv.Honest = F, split.Bucket = F,
```

```
+                     xval = 10, cv.alpha = 0.5, propensity = 0.5, cp = 0)
> rpart.plot(fit)
```



Then we do prunning to minize the cross validaiton error in `cptable`.

```
> opcp <- fit$cptable[, 1][which.min(fit$cptable[,4])]
> opfit <- prune(fit, cp = opcp)
> rpart.plot(opfit)
```



References

[1] Susan Athey and Guido Imbens. Machine learning methods for estimating heterogeneous causal effects. *arXiv preprint arXiv:1504.01132*, 2015.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, , and C.J Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Ca, 1983.