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

Susan Athey
Guido Imbens
Yanyang Kong

April 26, 2016

Contents

1 Introduction **2**

2 Notation **2**

3 Building Causal Trees **3**

3.1 Splitting rules . . . 3

3.1.1 Transformed Outcome Trees (**TOT**) . . . 4

3.1.2 Causal Trees (**CT**) . . . 4

3.1.3 Fit-based Trees (**fit**) . . . 5

3.1.4 Squared T-statistic Trees (**tstats**) . . . 5

3.2 Discrete splitting . . . 6

3.3 Example . . . 6

4 Cross Validation and Pruning **7**

4.1 Cross validation options . . . 7

4.1.1 TOT . . . 7

4.1.2 CT . . . 7

4.1.3 fit . . . 7

4.1.4 matching . . . 7

4.2 Example . . . 7

5 Honest Estimation **8**

1 Introduction

This document is a brief introduction of `causalTree` package, which is indended to give a short overview of the `causalTree` function and the `honest.causalTree` function, which implement the methods from *Recursive Partitioning for Heterogeneous Causal Effects* [1].

The `causalTree` function builds a regression model and returns an `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.

Following `rpart`, in the first stage, the tree is grown from the root node based on a specified splitting rule. In each node, the data in a leaf will be split into two groups to best minimize the risk function. Next, in the left sub-node and right sub-node, the splitting routine will be applied separately and so on recursively until no improvements can made, or until some limits are reached (e.g. the routine will stop if it cannot make splits that have at least `minsize` of treated observations and `minsize` control observations in each terminal node.)

In the second stage, the tree will be pruned using a specified cross-validation method, where the cross-validation penalty parameter penalizes the number of nodes in the tree. The leaves to be pruned are selected according to the risk function calculated while the tree is built.

The `causalTree` package incorporates an additional function not included in `rpart`, which is honest re-estimation `honest.causalTree` of causal effects. Honest here means that we estimate causal effects in the leaves of a given tree on an independent estimation sample rather than the data used to build and cross-validate the tree. The user first builds the tree with `causalTree`, specifying the training data for building the tree, and then passes the tree object as well as the estimation sample data into `honest.causalTree`, which replaces the leaf estimates from the input tree with new estimates in each leaf, calculated on the estimation sample.

2 Notation

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.	

$\mathcal{S}$ — a data sample drawn from data sample population,
$\mathcal{S}^{\text{tr}}$ denotes a training sample,
$\mathcal{S}^{\text{te}}$ denotes a test sample,
$\mathcal{S}^{\text{est}}$ denotes an estimation sample.
$\mathcal{S}_{\text{treat}}$ and $\mathcal{S}_{\text{control}}$ denote the subsamples of treated and control units.

Π — a partitioning tree $\Pi = \{\ell_1, \ldots, \ell_{\#(\Pi)}\}$ with $\cup_{j=1}^{\#(\Pi)} \ell_j = \mathbb{X}$ corresponds to a partitioning of the feature space the feature sapce $\mathbb{X}$, with $\#(\Pi)$ the number of elements in the partition.

$\ell(x; \Pi)$ — the leaf $\ell \in \Pi$ such that $x \in \ell$.

$\tau(\ell)$ — $l = 1, 2, ..., k$ causal effect or treatment effect in leaf ℓ.

p — marginal treatment probability, $p = \text{pr}(W_i = 1)$.

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 $(Y_i^{\text{obs}}, X_i, W_i)$, given a tree Π, the population average outcome is

$$\mu(w, x; \Pi) \equiv \mathbb{E}\left[Y_i(w) \middle| X_i \in \ell(x; \Pi) \right],$$

and its average causal effect is

$$\tau(x; \Pi) \equiv \mathbb{E}\left[Y_i(1) - Y_i(0) \middle| X_i \in \ell(x; \Pi) \right].$$

the estimated outcome is

$$\hat{\mu}(w, x; \mathcal{S}, \Pi) \equiv \frac{1}{\#(\{i \in \mathcal{S}_w : X_i \in \ell(x; \Pi)\})} \sum_{i \in \mathcal{S}_w : X_i \in \ell(x; \Pi)} Y_i^{\text{obs}},$$

the estimated causal effect is the difference of treated mean and control mean in the leaf l where it belongs,

$$\hat{\tau}(x; \mathcal{S}, \Pi) \equiv \tau(\ell) = \hat{\mu}(1, x; \mathcal{S}, \Pi) - \hat{\mu}(0, x; \mathcal{S}, \Pi).$$

3.1.1 Transformed Outcome Trees (TOT)

We first define the transformed outcome as

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

where $p = N_{\text{treat}}/N$ is the reatment probability, and

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

In **TOT** splitting rule, the risk function is given by

$$\widehat{\text{MSE}}(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \left\{ (Y_i^* - \hat{\tau}(X_i; \mathcal{S}^{\text{tr}}, \Pi))^2 - Y_i^{*2} \right\}$$

Note that the paper [1] envisions that treatment effects would be estimated by taking the mean of Y_i^* within a leaf, but points out that this is inefficient because the treated fraction in a leaf may differ from the population proportion due to sampling variation. Thus, our package uses $\hat{\tau}$ instead. The `rpart` package can be used off-the-shelf (applied with Y_i^* as the outcome) to implement the method precisely as described in [1].

3.1.2 Causal Trees (CT)

In causal trees splitting rule, we have two versions, adaptive verison, denoted as **CT-A**, and honest version, **CT-H**.
For **CT-A**, we use $\widehat{\text{MSE}}_\tau(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi)$ as the objective risk function, and

$$-\widehat{\text{MSE}}_\tau(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi).$$

For **CT-H**, the honest version, the splitting objective risk function is $\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, \Pi)$, and

$$\begin{aligned} -\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, \Pi) = {} & \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi) \\ & - \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)}{1-p} \right). \end{aligned}$$

where $S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)$ is the within-leaf variance on outcome Y for $\mathcal{S}^{\text{tr}}_{\text{control}}$ in leaf ℓ, and $S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)$ is the counter part for $\mathcal{S}^{\text{tr}}_{\text{treat}}$.

In our package we incorporate an additional parameter `split.alpha` $= \alpha \in (0,1)$ as a parameter to adjust the proportion of $\widehat{\text{MSE}}$ and the varaince term in $\widehat{\text{EMSE}}$.

$$
\begin{aligned}
-\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, \Pi, \alpha) = \alpha \cdot \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
- (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)}{p} + \frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)}{1-p} \right)
\end{aligned}
$$

3.1.3 Fit-based Trees (fit)

In fit-based splitting rule, we decide at what value of the feature to split based on the goodness-of-fit of the outcome rather than the treatment effect. As **CT**, there are two versions of **fit**, namely adaptive version **fit-A** and honest version **fit-H**.
For **fit-A**, the objective risk function in splitting is

$$
\widehat{\text{MSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{tr}}, \Pi) = \sum_{i \in \mathcal{S}^{\text{tr}}} \left\{ (Y_i - \hat{\mu}_w(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi))^2 - Y_i^2 \right\}
$$

For **fit-H**, the honest version, the risk function is $\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, \Pi)$,

$$
\begin{aligned}
-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, \Pi) = \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\mu}^2_w(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
- \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell) \right),
\end{aligned}
$$

where $S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell)$ is the within-leaf variance on outcome Y for $\mathcal{S}^{\text{tr}}_{\text{control}}$ in leaf ℓ, and $S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell)$ is the counter part for $\mathcal{S}^{\text{tr}}_{\text{treat}}$.
Also like **CT**, we have adjusted honest verison for $\widehat{\text{EMSE}}_{\mu,W}$ using `split.alpha`,

$$
\begin{aligned}
-\widehat{\text{EMSE}}_{\mu,W}(\mathcal{S}^{\text{tr}}, \Pi, \alpha) = \alpha \cdot \frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\mu}^2_w(W_i, X_i; \mathcal{S}^{\text{tr}}, \Pi) \\
- (1-\alpha) \cdot \left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}} \right) \cdot \sum_{\ell \in \Pi} \left(S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}(\ell) + S^2_{\mathcal{S}^{\text{tr}}_{\text{control}}}(\ell) \right),
\end{aligned}
$$

3.1.4 Squared T-statistic Trees (tstats)

In sqaured t-statistic trees, we consider the splits with the largest value for square of the t-statistic for testing the null hypothesis that the average treatment effect is the same in the two potential leaves. Denote the left leaf as L and right leaf as R, the square of the t-statistic is

$$T^2 \equiv \frac{((\overline{Y}_{L1} - \overline{Y}_{L0}) - (\overline{Y}_{R1} - \overline{Y}_{R0}))^2}{S_{L1}^2/N_{L1} + S_{L0}^2/N_{L0} + S_{R1}^2/N_{R1} + S_{R0}^2/N_{R0}},$$

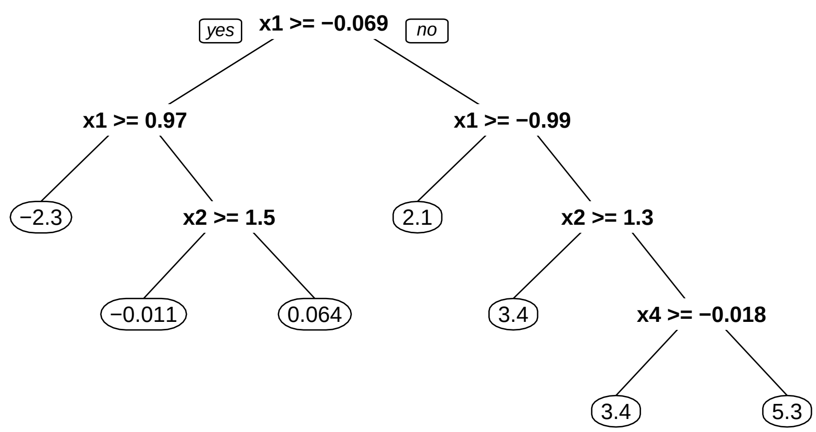
where $S_{\ell,w}^2$ is the conditional within treatment group sample variance given the split.

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

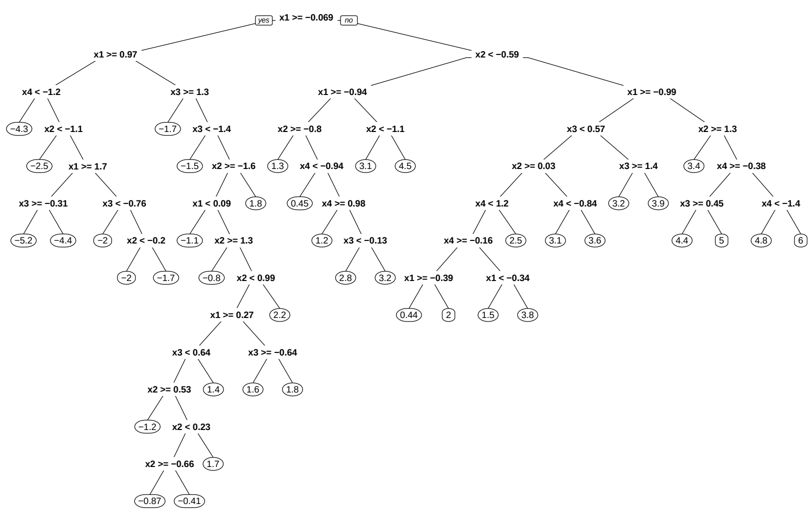
4.1.2 CT

4.1.3 fit

4.1.4 matching

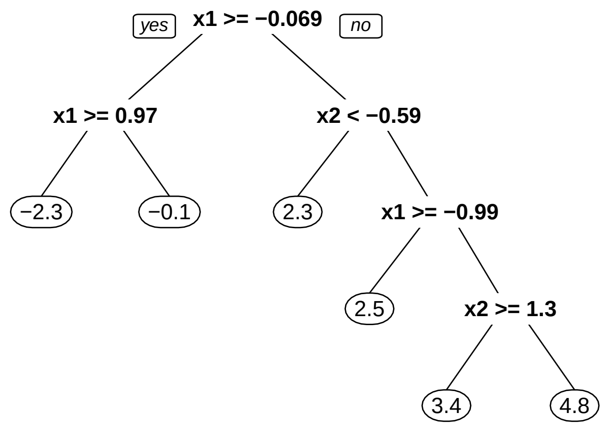
4.2 Example

```
> 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)
```



5 Honest Estimation

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.