ML in Aid of Estimation: Part II

Trees and CATE

Itamar Caspi

May 12, 2019 (updated: 2019-05-12)

Replicating this presentation

Use the pacman package to install and load packages:

```
if (!require("pacman"))
  install.packages("pacman")

pacman::p_load(tidyverse,
               tidymodels,
               rpart.plot,
               broom,
               knitr,
               xaringan,
               RefManageR)
```

Outline

1. Heterogenous Treatment Effects (HTE)
2. Challenges in Estimating HTE
3. Introducing Causal Trees (and Forests)
4. Empirical Illustration using `causalTree`

Heterogeneous Treatment Effects

Treatment and potential outcomes (Rubin, 1974, 1977)

- Treatment

- Potential outcomes

- Observed outcome: Under the Stable Unit Treatment Value Assumption (SUTVA), The realization of unit 's outcome is

- Individual treatment effect: The difference between unit 's potential outcomes:

Random treatment assignment

throughout this lecture, we assume that the treatments are randomly assigned. This means entails that is *independent* of potential outcomes, namely

Recall that randomized control trials (RCTs) enables us to estimate ATE using the average difference in outcomes by treatment status:

and its sample counterpart

i.e., the difference in average outcomes of the treatment and control groups is an unbiased estimate

Why should we we care about treatment effect heterogeneity?

- Typically there is reason to believe that a treatment might affect different individuals in different ways, e.g.,
    - Young subjects might respond better to a medicine
    - Short-term unemployed might respond better to job-raining programs
- In turn, better knowledge about treatment effect heterogeneity enables better treatment allocation:
    - Targeting treatment for those most likely to benefit from it.

Defining treatment effect heterogeneity

Recall the definition of ATE

Conditional treatment effect (CATE) is defined as

where is some specific value of or some range of values (a subspace of the feature space).

Challenges in Estimating HTE

"Moving the goalpost"

- Conditional average treatment effects (CATE) can be viewed as a compromise between ATE and personalized treatment effects.
- CATEs are ATEs for specific subgroup of individuals, where subgroups are classified based on the 's. Formally,

were now, is some partition of the features space .

For example, might represent the subgroup of individuals below 18 years old who weight more than 75 kg.

Estimating CATE using linear regression

The most common approach: Estimate the *best linear projection* (BLP) for while including interaction terms between the treatment and the set of features.

For example, for a binary treatment and a single feature , estimate the following regression by OLS:

The coefficient is the interaction effect and is interpreted as the difference between ATE and the effect of among individuals with .

REMARK: The parameter has a causal interpretation only when is randomly assigned.

Potential problems with the BLP approach

1. The above solution is infeasible when the number of attributes and interaction terms is large with respect to the number of observations.

2. Lasso can be used when , but can suffer from omitted variable bias (e.g., Lasso might drop some of the main effects).

Bias-variance trad-off in heterogeneous treatment effects

- Ideally, we would like to know "personalized" treatment effects, i.e., the effect of treatment on an individual with .
- Roughly speaking, the more personalized we get, the less biased is our estimate
- However, the more personalized we get, the more noisy is our estimate

Introducing Causal Trees (and Forests)

Notation: Data

Data

- observed outcome for individual .
- individual attributes vector.
- binary treatment indicator .

Sample

- the sample
- training sample
- test sample
- estimation sample
- treatment group
- control group

Observations

- \- total number of observations
- \- size of the training sample
- \- size of the the test sample
- \- size of the the estimation sample

Notation: Trees and CATE

Tree

- - attributes space
- - a partitioned tree
- - number of partitions
- - a leaf of such that
- a leaf such that

Treatment

- - treatment effect in leaf
- - marginal treatment probability,

.

What if we could observe ?

Say that we have data on and for .

Our task is to provide an out-of-sample prediction of for an individual with equals to some .

A naive approach would be to fit a regression tree to the data, where splits are based on in-sample fit

$$—$$

and regularization (pruning) on cross validation.

Causal tree (Athey and Imbens, PNAS 2016)

GOAL: Estimate heterogeneous treatment effects (CATE) .

THE BASIC IDEA: use a regression tree to form a partition of the attributes space .

CHALLENGES:

1. Conventional trees split leaves based on . We are interested in , which is unobserved.
2. What is the regularization criteria?
3. How to form confidence intervals?

SOLUTIONS:

1. Split tree based the heterogeneity and accuracy of .
2. Regularize based on treatment effect heterogeneity and accuracy within leaves.
3. Use sample splitting: Build tree on one sample and estimate CATE on a different and independent sample.

The naive approach

Use of-the-shelf CART to

1. Estimate two trees to predict outcomes , one for each subsample of treated and control.
2. Estimate a single tree for , and focus on splits in .

PROBLEM: The above naive approaches (tree construction and cross-validation) are optimized for outcome heterogeneity and not treatment heterogeneity. Implicitly relies on the assumption that treatment is highly correlated with the 's.

Approach #1: Transformed outcome trees (TOT)

Suppose we have an RCT with probability of receiving the treatment = 50%. Define

Then, is an unbiased estimate for individual 's .

PROOF: First, note that since we're in a 50-50 RCT,

where the expectation is with respect to the *probability of being treated*. similarly,

Non 50-50 assignment

More generally, if the probability of treatment assignment is given by , then

In observational studies, can be estimated based on the 's, i.e., use instead of setting a constant for all .

Once is defined, we can proceed with of-the-shelf tree methods for prediction:

1. Use a conventional algorithm (e.g., `rpart`) to fit a tree to predict .
2. Use the mean of within each leaf as the estimate for .

Problems with the TOT approach

PROBLEM: The TOT approach, CATE is estimated as the average within each leaf. and not as the difference in average outcome between the treatment and control groups.

EXAMPLE: in a leaf with 7 treated and 10 untreated, will be the average of , for .

What we really want is the average of minus the average of .

(NOTE: As we will discuss later, the `causalTree` package estimates within each leaf instead of .)

An aside: Sample splitting and honest estimation

sample splitting: divide the data in half, compute the sequence of models on one half and then evaluate their significance on the other half.

COST: this can lead to a significant loss of power,unless the sample size is large.

BENEFIT: Valid inference (independent subsamples).

In the context of causal trees, sample splitting amounts to constructing a tree using the training sample and estimating the effect using .

Approach #2: Causal tree (CT)

Solution: Define as the ATE within the leaf.

Athey and Imbens consider two splitting rules:

1. Adaptive causal tree (CT-A):

In words: perform split if it *increases* treatment effect heterogeneity within sample.

Approach #2: Causal tree (CT)

1. Honest causal tree (CT-A) which uses sample splitting:

where is the within-leaf variance on outcome for control in leaf , and is the counter part for treat.

In words: perform split if it *increases* treatment effect heterogeneity *and* reduces the uncertainty about the estimated effect.

Additional splitting rules

Athey and Imbens (2016) consider two additional splitting rules:

1. Fit based trees: split is based on the goodness-of-fit of the *outcome*, where fitting takes into account .
2. Squared -statistic trees: split according to largest value the square of the -statistic for testing the null hypothesis that the average treatment effect is the same in the two potential leaves.

See Athey and Imbens (2016) for more details.

Cross-validation and pruning

- Cross validation in causal trees is based on the out-of-sample counterpart of the goodness-of-fit rule used for constructing the tree.
- In particular, the training sample is split to training and validation sets and pruning the tree is constructed based on and validated using .

A summery of the causal tree algorithm

1. Randomly split the sample in half to form a training sample and an estimation sample .

2. Using just , grow a tree, where each split is based on a criteria that aims to maximize:

    - how much the treatment effect estimates *vary* across the two resulting subgroups (maximize treatment heterogeneity)
    - how *accurate* these estimates are (minimize estimate variance).

3. Using just , calculate within each terminal leaf .

Notes on the implementation of causal trees

- The causal tree algorithm is implemented in the `causalTree` package (Athey).
- The user is required to select
  - `minsize`: the minimum number of treatment and control observations in each leaf.
  - `bucketNum` and `bucketMax`: used to guarantee that when we shift from one split point to the next, we add both treatment and control observations, leading to a smoother estimate of the goodness of fit function as a function of the split point.

Extension: Causal forests (Wager and Athey, JASA 2018)

Causal Forests: As in the case of predictive trees, an individual causal tree can be noisy. We can reduce variance by using forests. Here is a sketch of the *causal forest* algorithm:

1. Draw a subsample without replacement from the observations in the dataset.
2. Randomly split in half to form a training sample and an estimation sample .
3. Using just , grow a tree , where each split is based on a criteria that aims to maximize:
    - how much the treatment effect estimates *vary* across the two resulting subgroups (maximize treatment heterogeneity)
    - how *accurate* these estimates are (minimize estimate variance).
4. Using just , calculate within each terminal leaf.
5. Return to the full sample and assign for each , based on where it is located in .
6. Repeat 1-5 times.
7. Define subject 's CATE as .

Notes on the implementation of causal forests

- The causal forest algorithm is implemented in the `grf` package (Tibshirani, Athey, and Wager).

- The user is required to select

  - number of trees.
  - subsample size.
  - minimum number of treatment and control observations in each leaf.
  - number of variables considered at each split (`mtry`).

- An excellent reference is Davis and Heller (2017) who apply causal forest to RCT that evaluates the impact of a summer jobs program on disadvantaged youth in Chicago.

Empirical Illustration using `causalTree`

The `causalTree` package

A description from the `causalTree` GitHub repository:

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

To install the package, run the following commands:

```r
# install.packages("devtools")
devtools::install_github("susanathey/causalTree")
```

To load the package:

```r
library(causalTree)
```

`experimentdatar` and the `social` dataset

A description from the `experimentdatar` GitHub repository:

> *"The `experimentdatar` data package contains publicly available datasets that were used in Susan Athey and Guido Imbens' course "Machine Learning and Econometrics" (AEA continuing Education, 2018). The datasets are conveniently packed for R users."*

You can install the *development* version from GitHub

```r
# install.packages("devtools")
devtools::install_github("itamarcaspi/experimentdatar")
```

Note: Downloading and installing the package may take a while due to its size.

Load the package

```r
library(experimentdatar)
```

The `social` dataset

The data is from Gerber, Green, and Larimer (2008)'s paper "Social Pressure and Voter Turnout: Evidence from a Large-Scale Field Experiment".

For this illustration, we will make use of the `social` dataset

```
data(social)
```

The following command will open a link to Gerber, Green, and Larimer (2008)'s paper

```
dataDetails("social")
```

Experimental design

A large sample of voters were *randomly assigned* to two groups:

- Treatment group that received a message stating that, after the election, the recent voting record of everyone on their households would be sent to their neighbors.
- Control group that did not receive any message.

This study seeks evidence for a "social pressure" effect on voters turnout.

The treatment and control messages

Dear Registered Voter:

DO YOUR CIVIC DUTY AND VOTE!

Why do so many people fail to vote? We've been talking about this problem for years, but it only seems to get worse.

The whole point of democracy is that citizens are active participants in government; that we have a voice in government. Your voice starts with your vote. On August 8, remember your rights and responsibilities as a citizen. Remember to vote.

DO YOUR CIVIC DUTY — VOTE!

Dear Registered Voter:

WHAT IF YOUR NEIGHBORS KNEW WHETHER YOU VOTED?

Why do so many people fail to vote? We've been talking about the problem for years, but it only seems to get worse. This year, we're taking a new approach. We're sending this mailing to you and your neighbors to publicize who does and does not vote.

The chart shows the names of some of your neighbors, showing which have voted in the past. After the August 8 election, we intend to mail an updated chart. You and your neighbors will all know who voted and who did not.

DO YOUR CIVIC DUTY — VOTE!

MAPLE DR	Aug 04	Nov 04	Aug 06
9995 JOSEPH JAMES SMITH	Voted	Voted	______
9995 JENNIFER KAY SMITH		Voted	______
9997 RICHARD B JACKSON		Voted	______
9999 KATHY MARIE JACKSON		Voted	______
9999 BRIAN JOSEPH JACKSON		Voted	______
9991 JENNIFER KAY THOMPSON		Voted	______

social: Outcome, treatment and attributes

- outcome_voted: Dummy where indicates voted in the August 2006
- treat_neighbors: Dummy where indicates *Neighbors mailing* treatment
- sex: male / female
- yob: Year of birth
- g2000: voted in the 2000 general
- g2002: voted in the 2002 general
- p2000: voted in the 2000 primary
- p2002: voted in the 2002 primary
- p2004: voted in the 2004 primary
- city: City index
- hh_size: Household size
- totalpopulation_estimate: City population
- percent_male: males in household
- median_age: Median age in household
- median_income: Median income in household
- percent_62yearsandover: of subjects of age higher than 62 yo
- percent_white: white in household
- percent_black: black in household
- percent_asian: Asian in household
- percent_hispanicorlatino: Hispanic or Latino in household
- employ_20to64: of employed subjects of age 20 to 64 yo
- highschool: having only high school degree
- bach_orhigher: having bachelor degree or higher

Data preprocessing

First, we define the outcome, treatment and other covariates

```r
Y <- "outcome_voted"

D <- "treat_neighbors"

X <- c("yob", "city", "hh_size",
       "totalpopulation_estimate",
       "percent_male", "median_age",
       "percent_62yearsandover",
       "percent_white", "percent_black",
       "percent_asian", "median_income",
       "employ_20to64", "highschool",
       "bach_orhigher", "percent_hispanicorlatino",
       "sex","g2000", "g2002", "p2000",
       "p2002", "p2004")
```

NOTE: The `social` dataset includes a much richer feature set. It includes additional treatments, as well as features.

Data wrangling

Load data and modelling packages

```
library(tidyverse)
library(tidymodels)
```

Set seed for replication and rename the outcome and treatment variables

```
df <- social %>%
  select(Y, D, X) %>%
  rename(Y = outcome_voted, W = treat_neighbors)
```

We will only use part of the sample to make things run faster

```
set.seed(1203)

df <- df %>%
  sample_n(50000)
```

Split the data to training, estimate, and test sets

Before we start, we need to split our sample to a training and estimaton sets, where training will be used to construct the tree and estimation for honest estimation of :

```
df_split <- initial_split(df, prop = 0.5)

df_tr   <- training(df_split)
df_est  <- testing(df_split)
```

Estimate causal tree

We now proceed to estimating the tree using the **CT-H** approach:

```
tree <- honest.causalTree("I(Y) ~ . - W",
                  data=df_tr,
                  treatment=df_tr$W,
                  est_data=df_est,
                  est_treatment=df_est$W,
                  split.Rule="CT",
                  split.Honest=TRUE,
                  split.Bucket=TRUE,
                  bucketNum=5,
                  bucketMax=100,
                  cv.option="CT",
                  cv.Honest=TRUE,
                  minsize=200,
                  split.alpha=0.5,
                  cv.alpha=0.5,
                  HonestSampleSize=nrow(df_est),
                  cp=0)
```

Prune the tree based on (honest) cross-validation

```
opcp   <- tree$cptable[, 1][which.min(tree$cptable[, 4])]
pruned_tree <- prune(tree, cp = opcp)
```

The estimated tree

Pruned tree

Assign each observation to a specific leaf

Form a tibble which holds the training and estimation samples

```
df_all <- tibble(sample = c("training",
                            "estimation"),
                  data = list(df_tr, df_est)
                 )
```

Assign each observation in the training and estimation samples to a leaf based on `tree`:

```
df_all_leaf <- df_all %>%
  mutate(leaf = map(data, ~ predict(pruned_tree,
                      newdata = .x,
                      type = "vector"))) %>%
  mutate(leaf = map(leaf, ~ round(.x, 3))) %>%
  mutate(leaf = map(leaf, ~ as.factor(.x))) %>%
  mutate(leaf = map(leaf, ~ enframe(.x, name = NULL, value = "leaf"))) %>%
  mutate(data = map2(data, leaf, ~ bind_cols(.x, .y)))
```

Estimate CATE using the causal tree

Use `lm()` with interaction terms, e.g.,

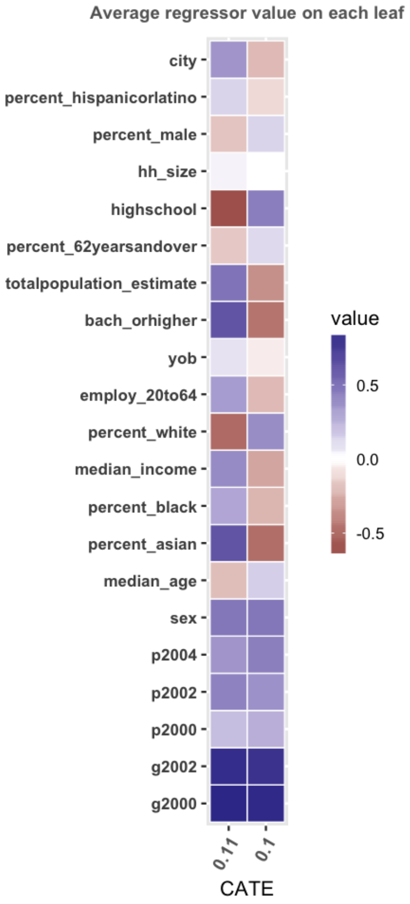
`lm(Y ~ leaf + W * leaf - W - 1)`

to estimate the average treatment effect within each leaf and to get confidence intervals:

```
df_all_lm  <- df_all_leaf %>%
  mutate(model = map(data, ~ lm(Y ~ leaf + W * leaf
                                 - W - 1, data = .x))) %>%
  mutate(tidy = map(model, broom::tidy, conf.int = TRUE)) %>%
  unnest(tidy)
```

Plot coefficients and confidence intervals

On the interpretation of causal trees



Source: https://drive.google.com/open?id=1FuF4_q4HCzbU_ImFoLW4r4Gop6A0YsO_

```
slides %>% end()
```

Source code

Selected references

Athey, S., & Imbens, G. (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27), 7353-7360.

Athey, S., Imbens, G. W., Kong, Y., & Ramachandra, V. (2016). An Introduction to Recursive Partitioning for Heterogeneous Causal Effects Estimation Using `causalTree` package. 1–15.

Davis, J.M. V & Heller, S.B., 2017. Using Causal Forests to Predict Treatment Heterogeneity : An Application to Summer Jobs. *American Economic Review: Papers & Proceedings*, 107(5), pp.546–550.

Wager, S., & Athey, S. (2018). Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523), 1228–1242.