Causal Forest

Causal forests are a causal inference learning method that are an extension of Random Forests. In random forests, the data is repeatedly split in order to minimize prediction error of an outcome variable. Causal forests are built similarly, except that instead of minimizing prediction error, data is split in order to maximize the difference across splits in the relationship between an outcome variable and a “treatment” variable. This is intended to uncover how treatment effects vary across a sample.

For more information, see Explicitly Optimizing on Causal Effects via the Causal Forest.

Keep in Mind

- Causal forests simply uncover heterogeneity in a causal effect, they do not by themselves make the effect causal. A standard causal forest must assume that the assignment to treatment is exogenous, as it might be in a randomized controlled trial. Some extensions of causal forest may allow for covariate adjustment or for instrumental variables. See your causal forest package’s documentation to see if it has an option for ways of identifying the causal effect when treatment is not exogenous such as conditional adjustment or “instrumental forest”.
- If using causal forest to estimate confidence intervals for the effects, in addition to the effects itself, it is recommended that you increase the number of trees generated considerably.

Also Consider

- Your intuition for how causal forest works can be based on a thorough understanding of Random Forests, for which materials are much more widely available.

Implementations

Python

The **econml** package from Microsoft provides a range of causal machine learning functions, including deep instrumental variables, doubly robust learning, double machine learning, and causal forests. As in the R example below, we will download some crime data and look at the effect of one variable (‘pctymle’, the % of young males, assumed to be exogenous) on another (‘crmrte’, the crime rate).

```
# Use "pip install econml" on the command line to install the package
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from econml.ortho_forest import ContinuousTreatmentOrthoForest as CausalForest

df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Crime.csv')

# Set the categorical variables:
cat_vars = ['year', 'region', 'smsa']
# Transform the categorical variables to dummies and add them back in
xf = pd.get_dummies(df[cat_vars])
df = pd.concat([df.drop(cat_vars, axis=1), xf], axis=1)
cat_var_dummy_names = list(xf.columns)

regressors = ['prbarr', 'prbconv', 'prbpris',
              'avgsen', 'polpc', 'density', 'taxpc',
              'pctmin', 'wcon']
# Add in the dummy names to the list of regressors
regressors = regressors + cat_var_dummy_names

# Split into train and test
train, test = train_test_split(df, test_size=0.2)

# Estimate causal forest
estimator = CausalForest(n_trees=100,
                         model_T=DecisionTreeRegressor(),
                         model_Y=DecisionTreeRegressor())
estimator.fit(train['crmrte'],
              train['pctymle'],
              train[regressors],
              inference='blb')
effects_train = estimator.effect(train[regressors])
effects_test = estimator.effect(test[regressors])
conf_intrvl = estimator.effect_interval(test[regressors])
```

R

The **grf** package has a `causal_forest` function that can be used to estimate causal forests. Additional functions afterwards can estimate, for example, the `average_treatment_effect()`. See `help(package='grf')` for more options.

```
# If necessary
# install.packages('grf')
library(grf)

# Get crime data from North Carolina
df <- read.csv('https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Crime.csv')

# It's not, but let's pretend that "percentage of young males" pctymle is exogenous
# and see how the effect of it on crmrte varies across the other measured covariates

# Make sure the data has no missing values. Here I'm dropping observations
# with missing values in any variable, but you can limit the data first to just
# variables used in analysis to only drop observations with missing values in those variables
df <- df[complete.cases(df),]

# Let's use training and holdout data
split <- sample(c(FALSE, TRUE), nrow(df), replace = TRUE)
df.train <- df[split,]
df.hold <- df[!split,]

# Isolate the "treatment" as a matrix
pctymle <- as.matrix(df.train$pctymle)

# Isolate the outcome as a matrix
crmrte <- as.matrix(df.train$crmrte)

# Use model.matrix to get our predictor matrix
# We might also consider adding interaction terms
X <- model.matrix(lm(crmrte ~ -1 + factor(year) + prbarr + prbconv + prbpris +
                       avgsen + polpc + density + taxpc + factor(region) + factor(smsa) +
                       pctmin + wcon, data = df.train))
```

```
# Estimate causal forest
cf <- causal_forest(X,crmrte,pctymle)

# Get predicted causal effects for each observation
effects <- predict(cf)$predictions

# And use holdout X's for prediction
X.hold <- model.matrix(lm(crmrte ~ -1 + factor(year) + prbarr + prbconv + prbpris +
                       avgsen + polpc + density + taxpc + factor(region) + factor(smsa) +
                       pctmin + wcon, data = df.hold))
# And get effects
effects.hold <- predict(cf, X.hold)$predictions

# Get standard errors for the holding data predictions - we probably should have set the num.trees
# option in causal_forest higher before doing this, perhaps to 5000.
SEs <- sqrt(predict(cf, X.hold, estimate.variance = TRUE)$variance.estimates)
```

Stata

The **MLRtime** package allows the `causal_forest` function in the R **grf** package to be run from inside of Stata. This does require that R be installed.

```
* If necessary, install MLRtime
* net install MLRtime, from("https://raw.githubusercontent.com/NickCH-K/MLRtime/master/")
* Then, before use, install R from R-project.org
* and run the MLRtimesetup function
* MLRtimesetup, go

* Start a fresh R session
rcall clear

* Get crime data from North Carolina
import delimited using "https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Crime.csv", clear

* Turn character variables numeric so we can use them
encode region, g(regionn)
encode smsa, g(smsan)
drop region smsa

* It's not, but let's pretend that "percentage of young males" pctymle is exogenous
* and see how the effect of it on crmrte varies across the other measured covariates

* Let's use training and holdout data by sending our holdout data to R with rcall
g split = runiform() > .5
preserve
* Keep the predictors from the holding data, send it over, so later we can make an X matrix to predict wit
keep if split == 0
keep year prbarr prbconv prbpris avgsen polpc density taxpc regionn smsan pctmin wcon
* R needs that data pre-processed! So using the same variables as in the main model, process the variable
fvrevar year prbarr prbconv prbpris avgsen polpc density taxpc i.regionn i.smsan pctmin wcon
keep `r(varlist)'
* Then send the data to R
rcall: df.hold <- st.data()
restore
* Now go back to just the training data

* Run causal_forest, storing the effect predictions for the training data in the "effects" variable
* the SEs of those effects in effectSE
* And the effects and SEs for the holdout data in matrices called effects_hold and effectSE_hold
causal_forest crmrte pctymle year prbarr prbconv prbpris avgsen polpc density taxpc i.regionn i.smsan pct

* Look at the holdout effects predicted
di "`r(effects_hold)'"
```