Enhancing Causal Models using Machine Learning

Elisa TOSETTI

Methods and Applications of Machine Learning

2023-2024

Outline of the lesson

Use Machine Learning for enhancing causal models:

- Meta learners: the S-, T-, X-learners
- Causal trees/causal forests
- Double/Debiased ML

Resources:

Kunzel et al (2018) Metalearners for estimating heterogeneous treatment effects using machine learning

Susan Athey an Guido Imbens (2016), Recursive partitioning for heterogeneous causal effects

Philipp Bach, Victor Chernozhukov, Malte S. Kurz, Martin Spindler, DoubleML – An Object-Oriented Implementation of Double Machine Learning in R

Notes

For the first two methods we need:

- Binary treatment variable D_i
- (Potentially high dimensional) covariates

The last method allows for both binary or continuous treatment variable as well as potentially high dimensional set of covariates

Potential outcome framework

Remember

$$\text{Potential outcome} = \begin{cases} y_{1i} \text{ if } D_i = 1 \\ y_{0i} \text{ if } D_i = 0 \end{cases}$$

When condition $\{y_{1i}, y_{0i}\} \perp D_i$ is violated, we say that we are in an **observational setting**

A confounding variable, also known as **confounder**, is a variable that affects both (i) The likelihood of receiving the treatment (ii) The outcome Y

Uncounfoundedness

Sometimes it is still possible to estimate the ATE under additional assumptions

One useful assumption is

$$\{y_{1i}, y_{0i}\} \perp D_i | X$$

we call this assumption **unconfoundeness**

It says that all possible sources of self-selection can be explained by the observable covariates, X

Under this case we obtain the CATE (conditional average treatment effect):

$$\tau(X) = E(y_{1i}|X) - E(y_{0i}|X)$$

Unconfoundedness

Under unconfoundedness we can write the CATE function as:

$$\tau(X) = E(y_{1i}|X) - E(y_{0i}|X)$$

$$= E(y_i|D_i = 1, X) - E(y_i|D_i = 0, X) = \mu_1(X) - \mu_0(X)$$

This representation is the starting point of several machine learning based estimation strategies of causal effects

In fact, any function μ can be used to calculate the CATE

Heterogeneous treatment effects

We want to obtain an unbiased estimate of $\mu_1(X)$ and $\mu_0(X)$

Note that $\mu_1(X)$ and $\mu_0(X)$ vary across units $\rightarrow$ We are interested in finding **treatment effect heterogeneity**:

- We want to know how sensitive the units are to the treatment
- This is useful in the case where we can't treat everyone and need to do some prioritization of the treatment, for example when you want to give discounts but have a limited budget
- Essentially, we need a data-driven partition of the X-space to capture heterogenous treatment effects
- Ideally, we would also like to have standard errors for $\hat{\tau}(X)$

Meta-learners

Meta learners are a simple way to leverage off-the-shelf predictive machine learning methods to estimate the CATE

Meta-learners are the result of combining supervised learning in a specific manner while **allowing the base learner to take any form**

Hence, meta-learners algorithm do not prescribe any particular ML method

We should use an approach that's sensible in the machine learning sense, i.e. striking a good bias/variance tradeoff or balancing under/overfitting

Idea developed by Kunzel et al (2018)

The S-learner

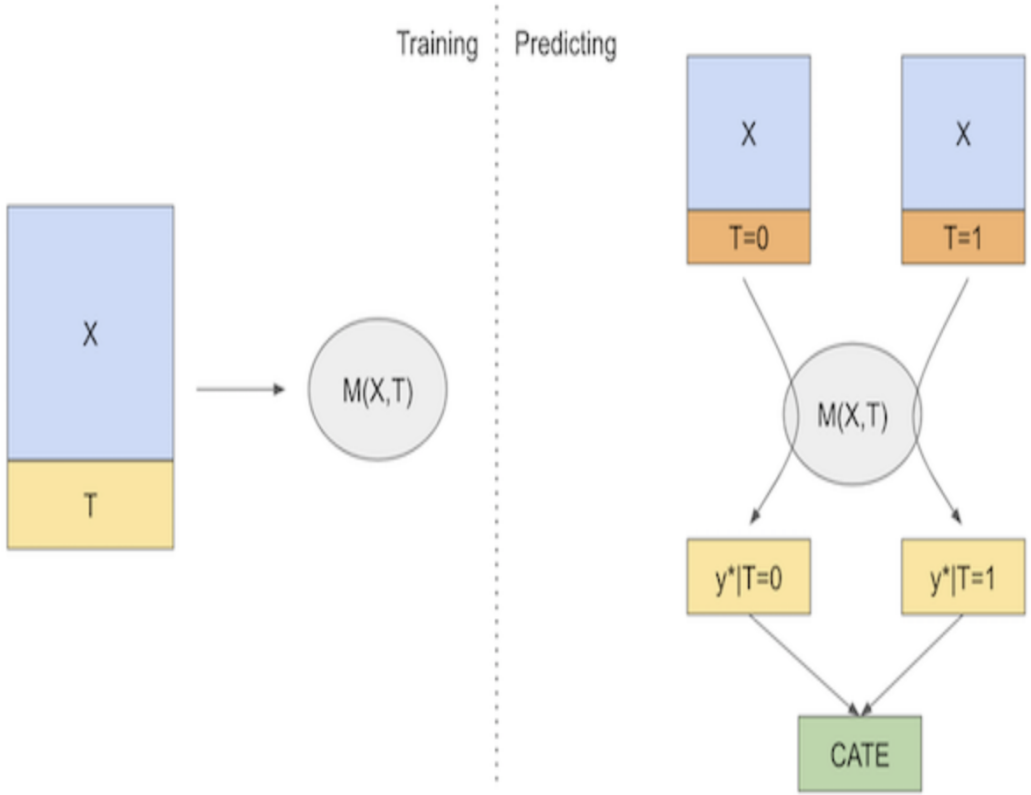
The S-learner takes only one step. Use any machine learning "base learner" for the following:

1. Estimate the combined response function $\mu(D, X) = E(Y|D, X)$. Let $\hat{\mu}(D, X)$ be our estimate for $\mu(D, X)$

The CATE estimator is given by $\hat{\tau}(x)^S = \hat{\mu}(1, X) - \hat{\mu}(0, X)$

With a regression tree learner, this was originally called the **single tree model**

The S-learner



The S-learner: an example

Consider the following data on an investment firm that wishes to test the effectiveness of sending a financial education email in making them invest more. The outcome variable is *converted* (invested more or not), the treatment variable is *em1* (educational email sent or not), *age*, *income*, *insurance* and *invested* are controllers.

age	income	insurance	invested	em1	converted
44.1	5483.80	6155.29	14294.81	0	0
39.8	2737.92	50069.40	7468.15	1	0
49.0	2712.51	5707.08	5095.65	0	1
39.7	2326.37	15657.97	6345.20	0	0
35.3	2787.26	27074.44	14114.86	1	0

The S-learner: an example

To estimate the S-learner, we simply include the treatment as a feature in the model that tries to predict the outcome Y. Using Gradient Boosting. . . .

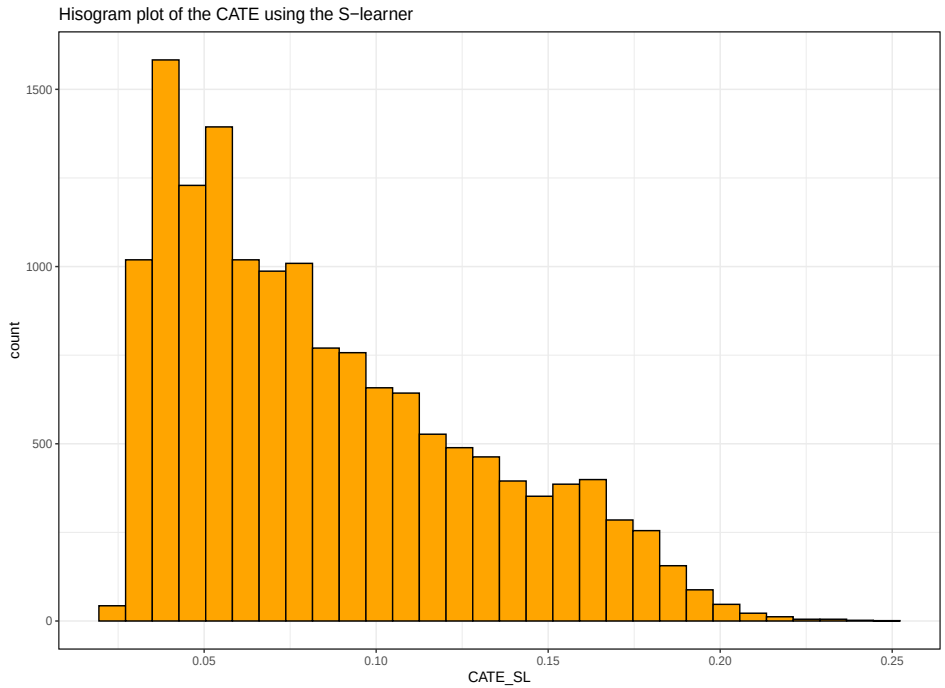
```
tf0 <- gbm(converted ~ em1 +  age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "bernoulli", data=train)
```

Prediction step:

```
X1<-test
X1$em1 <- 1 # create fictional data where all individuals have em1 = 1
X0<-test
X0$em1 <- 0 # create fictional data where all individuals have em1 = 0
test$CATE_SL <- predict(tf0,  newdata = X1, type="response", n.trees = 100)
```

The S-learner: an example

. . . .which allows to estimate the CATE:



The S-learner: an example

. . . and the correlation of the CATE with included regressors is:

```
##              CATE_SL   age income insurance invested
## CATE_SL         1.00  0.32   0.69     -0.07     0.77
## age             0.32  1.00   0.10     -0.01     0.16
## income          0.69  0.10   1.00     -0.01     0.81
## insurance      -0.07 -0.01  -0.01      1.00    -0.01
## invested        0.77  0.16   0.81     -0.01     1.00
```

The S-learner

In general, one major disadvantage of the S-learner is that it tends to bias the treatment effect towards zero

Since the S-learner employs what is usually a regularized machine learning model, that regularization can restrict the estimated treatment effect

Even worse, the S-learner may end up ignoring the treatment indicator D altogether if the treatment effect is small relative to the part of the outcome that is predictable from the other covariates, X

In this case, when the tree does not split on D, the S-learner would just give $\hat{\tau}(X)^S = 0$

The T-learner

The T-learner tries to solve the problem of discarding the treatment entirely by forcing the learner to first split on it

Instead of using a single model, we will use one model per treatment level

The T-learner takes two steps. Use any machine learning "base learner" for the following:

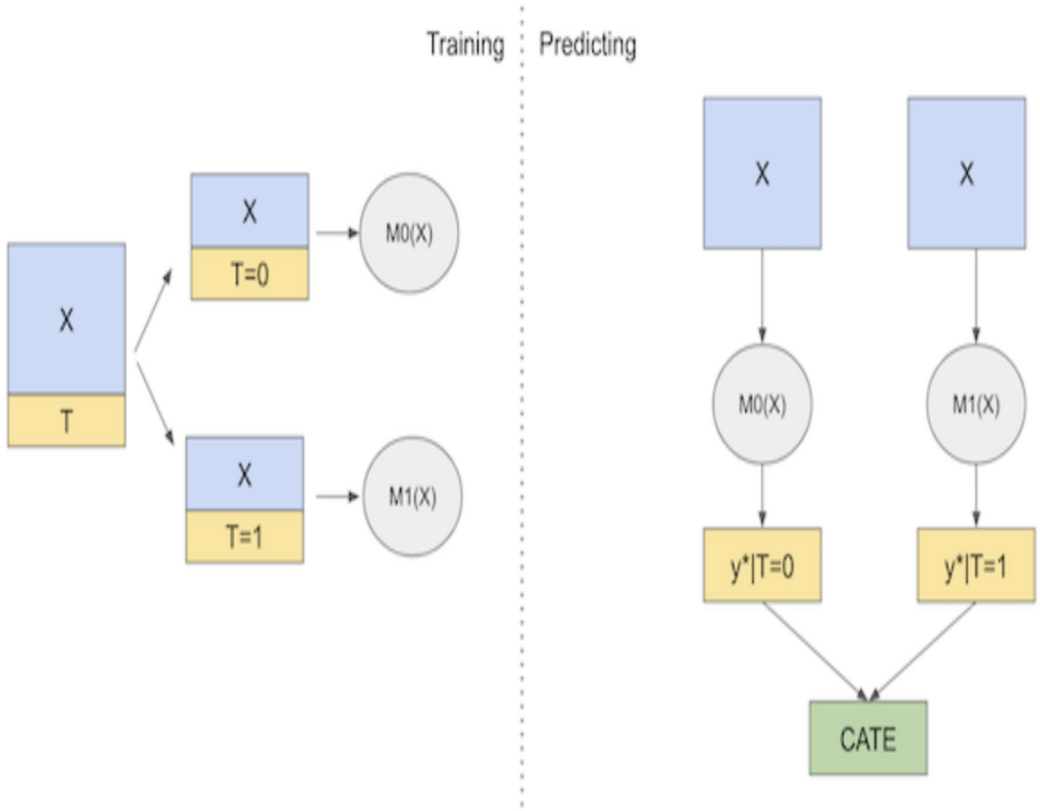
1. Estimate control response function $\mu_0(X) = E(y_{0i}|X)$ using observations in the control group
2. Estimate treatment response function $\mu_1(X) = E(y_{1i}|X)$ using observations in the treatment group

The CATE estimator is given by

$$\hat{\tau}(X)^T = \hat{\mu}_1(X) - \hat{\mu}_0(X)$$

Estimation was originally done with tree-based learners and called the **two trees model**

The T-learner



The T-learner: an example

Going back to the example on the investment firm, we apply the T-learner using the GBM learner....

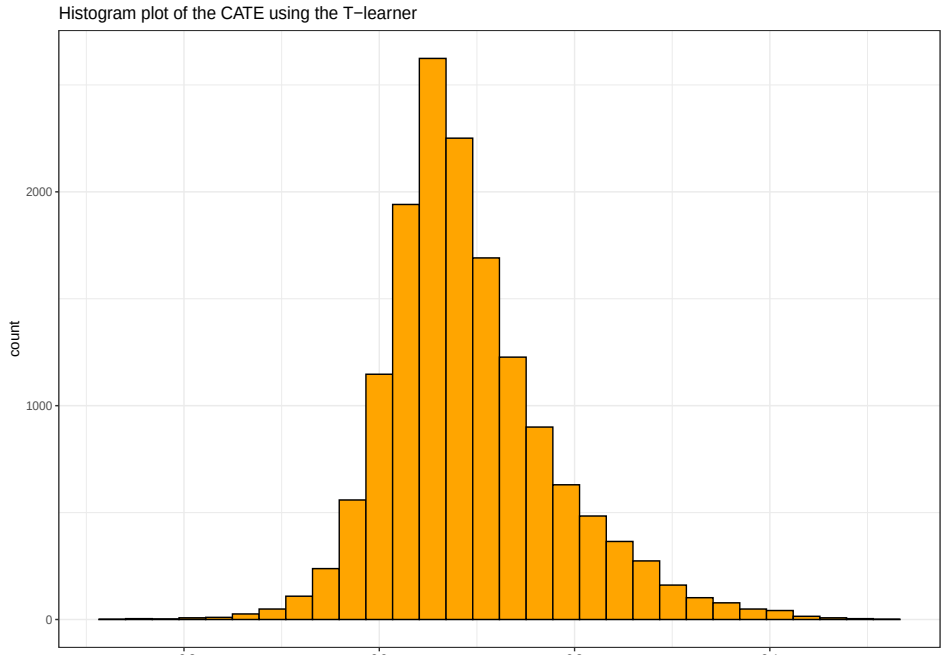
```
tf1 <- gbm(converted ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "bernoulli", data=train[train$em1==1,])
tf0 <- gbm(converted ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "bernoulli", data=train[train$em1==0,])
```

Notice that I have estimated two different models, one for the treatment and one for the control group

The T-learner: an example

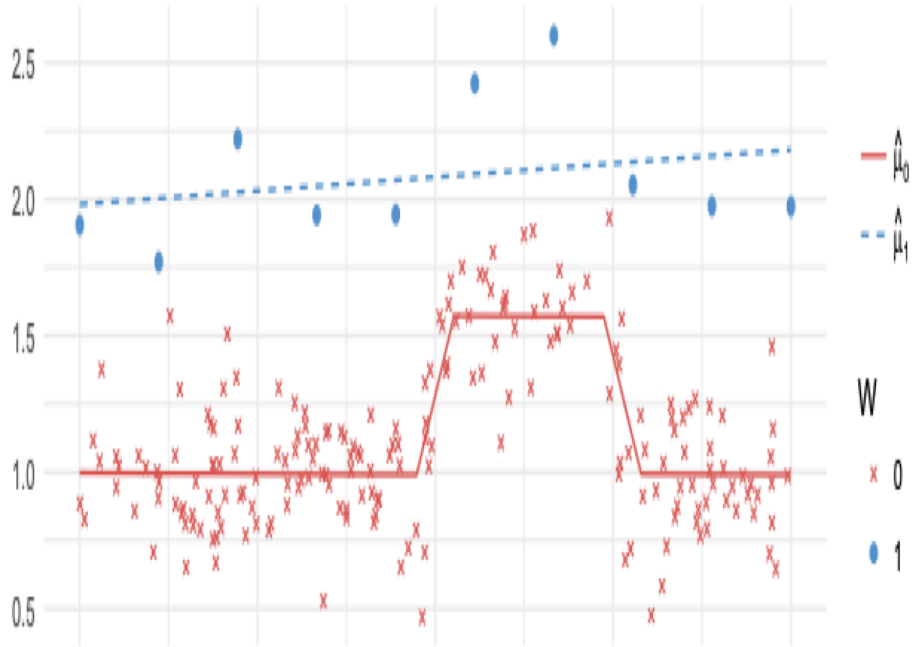
... prediction step and estimation of the CATE....

```
X1<-test
X1$em1 <- 1
X0<-test
X0$em1 <- 0
test$CATE_2L<-predict(tf1,  newdata = X1, type="response")-
  predict(tf0,  newdata = X0, type="response")
```



The T-learner

The T-Learner avoids the problem of not picking up on a weak treatment variable, but it can still suffer from regularization bias:



What happens here is that the model for the untreated can pick up the non linearity, but the model for the treated cannot, because we've used regularization to deal with a small sample size

To solve this problem, we can use an X-learner, proposed by Kunzela et

The X-learner

The X-Learner has two stages and a propensity score model

The first stage is identical to the T-learner. First, we split the samples into treated and untreated and fit a ML model for the treated and for control

For the second stage, we impute the treatment effect for the control using the model for the untreated, and impute the treatment effect for the treated using the model for the treated

Then, we fit two more models to predict those effects

So we have one model that is wrong because we've impute the treatment effects wrongly and another model that is correct because we've imputed those values correctly. Now, we need a way to combine the two in a way that gives more weight to the correct model

Here is where the propensity score model comes to play

The X-learner

To sum up, with the X-learner use any machine learning "base learner" for the following:

1. Estimate control response function $\mu_0(X) = E(Y_{0i}|X)$ and $\mu_1(X) = E(Y_{1i}|X)$ using observations in the control group and treatment group
2. Impute the treatment effects for the individuals in the treated group, based on the control-outcome estimator and the treatment effects for the individuals in the control group, based on the treatment-outcome estimator. That is, we define the **imputed treatment effects**:

$$\hat{\tau}_1(X) = Y(D = 1) - \hat{\mu}_0(X)$$

$$\hat{\tau}_0(X) = \hat{\mu}_1(X) - Y(D = 0)$$

3. Use any supervised learning to estimate $\hat{\tau}(X)$ in two ways: using the imputed treatment effects as the response variable in the treatment group to obtain $\hat{\mu}_{\tau_1}(X)$, and similarly in the control group $\hat{\mu}_{\tau_0}(X)$

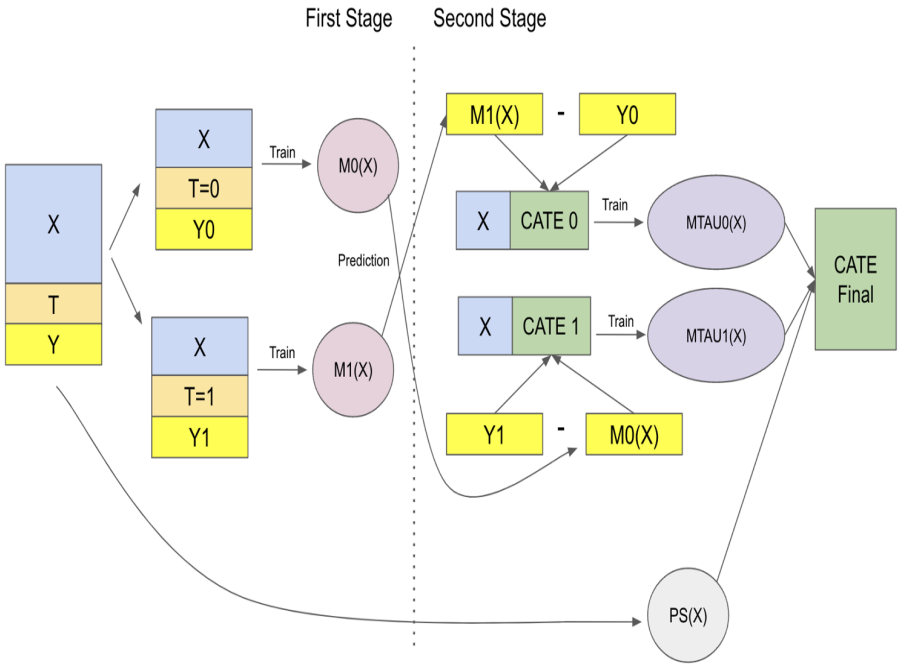
The X-learner

Let $\hat{e}(X)$ be the propensity score model, we can combine the two second stage models as follows:

$$\hat{\tau}(X)^X = \hat{e}(X)\hat{\tau}_0(X) + (1 - \hat{e}(X))\hat{\tau}_1(X)$$

If there are very few treated units, $\hat{e}(X)$ is very small and this will give a very small weight to the wrong model, $\hat{\tau}_0(X)$

The X-learner



The X-learner: an example

I our investment firm example, first, we need to estimate control response function $\mu_0(X) = E(Y_{0i}|X)$ and $\mu_1(X) = E(Y_{1i}|X)$ using observations in the control group and treatment group, and then we calculate the imputed treatment effects:

```
tf1 <- gbm(converted ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "bernoulli", data=train[train$em1==1,])

tf0 <- gbm(converted ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "bernoulli",  data=train[train$em1==0,])

tau1hat <- train$converted[train$em1==1] -
  predict(tf0, train[train$em1==1,], type="response", n.trees = 100)
tau0hat <-predict(tf1, train[train$em1==0,],type="response", n.trees = 100)
  train$converted[train$em1==0]
```

The X-learner: an example

Hence, we use the GBM to estimate $\hat{\tau}(X)$ in two ways: using the imputed treatment effects as the response variable in the treatment group to obtain $\hat{\mu}_{\tau_1}(X)$, and similarly in the control group $\hat{\mu}_{\tau_0}(X)$:

```
xf1 <- gbm(tau1hat ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "gaussian", data=train[train$em1==1,])
xf0 <- gbm(tau0hat ~   age + income + insurance + invested,
           interaction.depth=3, n.minobsinnode=30,
           distribution = "gaussian", data=train[train$em1==0,])
```

The X-learner: an example

We calculate predicted values. . . .

```
# Predict treatment effects
xf.preds.0 <- predict(xf0, train ,  n.trees = 100)
xf.preds.1 <- predict(xf1, train ,  n.trees = 100)
```

The X-learner: an example

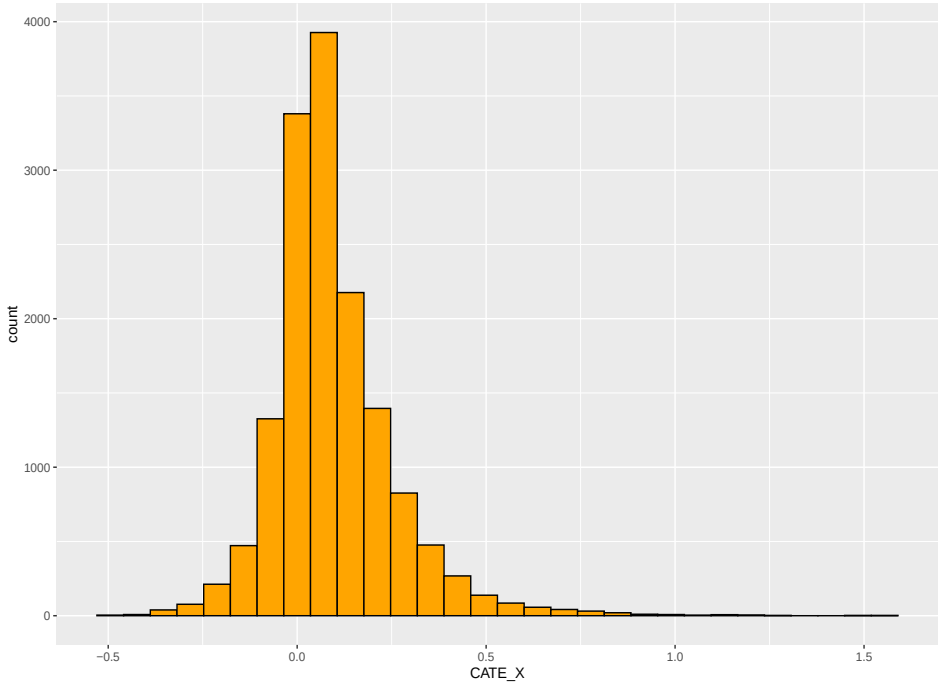
Finally, we estimate the propensity score

```
propf <- gbm(em1 ~ age + income + insurance + invested,
             interaction.depth=3, n.minobsinnode=30,
             distribution = "bernoulli", data=train)
ehat <- predict(propf, n.trees = 100)
```

. . . and compute the X-learner prediction by averaging the the two predicted CATEs using the propensity scores as weights:

```
test$CATE_X <- (1 - ehat) * xf.preds.1 + ehat * xf.preds.0
```

The X-learner: an example



Note: we could calculate the above also for the testing sample

Causal trees

As before we want to estimate the conditional average treatment effect

$$\tau(X) = E(y_{1i} - y_{0i}|X)$$

under the unconfoundedness assumption on potential outcomes

Causal trees provide a data-driven approach to partitioning the data into subgroups that **differ by the magnitude of their treatment effects**

Like decision trees, which partition the covariate space by finding subgroups with similar outcomes, causal trees find subgroups with **similar treatment effects**

Causal trees

Due to the fundamental problem of causal inference, directly training machine learning methods on the difference $y_i(1) - y_i(0)$ is not possible, as we do not observe both outcomes for any individual unit

Thus, Athey and Imbens (2016) propose to maximize a criterion function that rewards a split that increases the variance of treatment effects across leaves and penalizes a split that increases within-leaf variance

The split is performed if it improves the criterion function, compared to no split. When no more splits can be done, the tree constructed based on the first subsample is defined

Using only the estimation sample, the treatment effect in each leaf, ℓ, is computed as $\hat{\tau}_\ell = \bar{y}(D = 1) - \bar{y}(D = 0)$, i.e., the mean outcome difference between the treated and control observations within a leaf

Honesty criterion

Honesty criterion: it consists of partitioning the training data into **two separate** samples used for

1. Grow the tree (i.e. choosing the splits) $\rightarrow$ Training sample S^{tr}
2. Estimate treatment effects within each leaf of the tree $\rightarrow$ Estimation sample S^{est}

$\rightarrow$ a statistical unit can only be used either to estimate τ or to decide how to build the model (e.g. where to place the splits in trees), *but not both*

Honesty leads to two desirable characteristics:

1. It reduces bias from overfitting
2. It makes the inference valid, since the asymptotic properties of the treatment effect estimates are the same as if the structure of the tree had been exogenously given

Honest target

For a given partition done on a sample S, Π^S, let $\hat{\mu}(X, S, \Pi^S)$ be an estimate of $\mu(X)$ on a particular sample S using the partition Π^S

For causal trees, we modify the objective function with two adjustments

$$\sum_{i \in S^{te}} \{(y_i - \hat{\mu}(X, S^{est}, \Pi^{tr})^2 - y_i^2\}$$

Note that:

- S^{te} is the testing sample
- We use the partition Π^{tr} calculated on the *training sample*
- $\hat{\mu}()$ is calculated on the *estimation sample*
- the last term is a normalisation term

How to grow a causal tree

1. Use training sample to grow trees using recursive tree splitting by minimizing the in-sample splitting criteria on the previous slides and penalizing complexity at different levels of λ
2. Cross-validate to find the best λ and get the corresponding tree
3. Use estimation sample to find unbiased estimates of $\hat{\tau}$ for each leaf

Causal trees: an example

I our investment firm example, I can estimate causal trees using the *honest.causalTree* function from the *causalTree* R package...

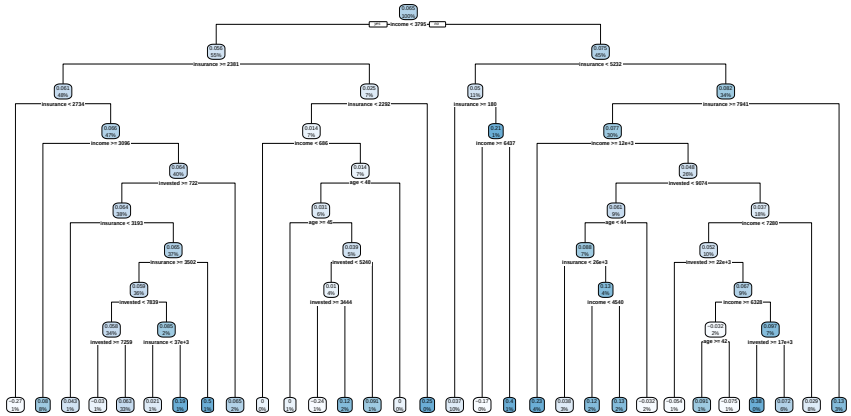
```
library("causalTree")
```

```
causal_tree <- honest.causalTree(converted ~
                    em1 +  age + income + insurance + invested,
            data=train,  # data for splitting
            treatment = train$em1, # treatment variable
            est_data = test,   # data for computing CATE
            est_treatment = test$em1, # treatment variable
            split.Rule = "CT",  # splitting rule: Causal Tree
            split.Honest = TRUE,  # we assume honest splitting
            cv.option = "CT")
```

```
## [1] 2
## [1] "CT"
```

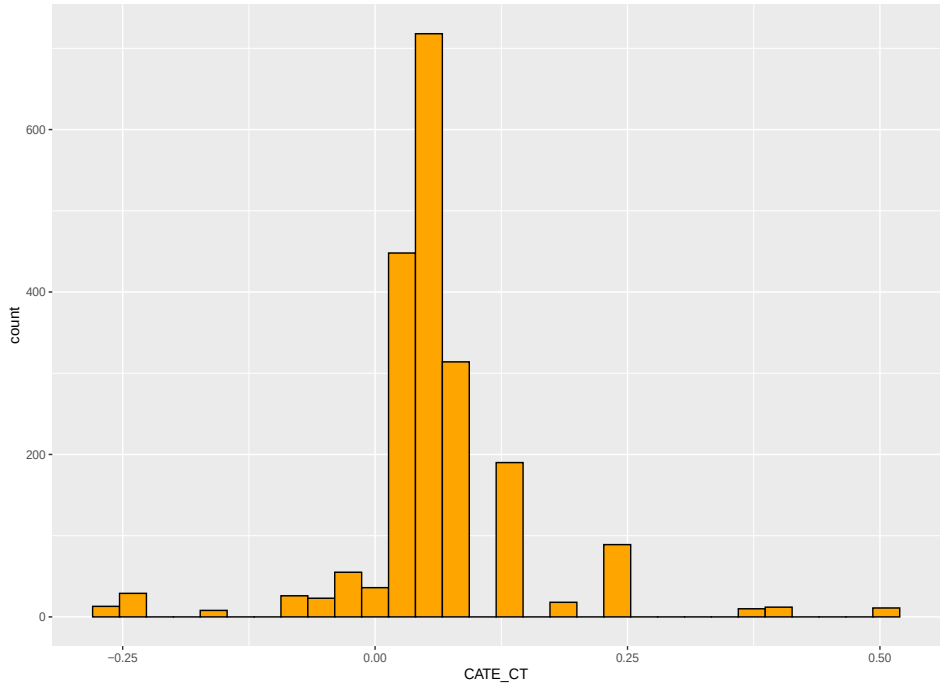
Causal trees: an example

. . . which leads to the estimation of the following best tree:



Causal trees: an example

. . . and estimation of the CATE. . .



Double/debiased machine learning

There is a rapidly growing recent literature on double machine learning, also called **double/debiased machine learning**

The approach is very general and encompasses a number of specific models

The approach relies on the results from the **Frisch, Waugh and Lovell (FWL)** theorem

Frisch, Waugh and Lovell were 20th century econometricians who noticed a very interesting feature of linear regression

Frisch-Waugh-Lovell

Suppose you have a linear regression model with a set of features X_1 and another set of features X_2. It is possible to obtain the model's parameters:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 \mathbf{x}_{1i} + \hat{\beta}_2 \mathbf{x}_{2i}$$

where $\hat{\beta}_1$ and $\hat{\beta}_2$ are row vectors

Suppose we are only interested in estimating β_1, while β_2 are some nuisance parameters

For example, $\mathbf{x}_{1i}$ is the treatment variable and $\mathbf{x}_{2i}$ are the confounders

Frisch-Waugh-Lovell theorem

It is possible to get the exact same $\hat{\beta}_1$ parameter by following three steps:

1. Regress y on the second set of features $\mathbf{x}_{2i}$, $\rightarrow$ obtain the residuals $\hat{y}_i^*$
2. Regress $\mathbf{x}_1$ on $\mathbf{x}_2$: $\rightarrow$ obtain the residuals $\hat{\boldsymbol{x}}_{1i}^*$
3. Regress the residuals of the outcome, $\hat{y}_i^*$, on the residuals of the features, $\hat{\boldsymbol{x}}_{1i}^*$

Under the FWL theorem the OLS coefficient on Step 3 is identical to OLS coefficient in multiple regression

Having the FWL theorem in mind, let's move to understanding the key ideas underlying Double/debiased ML

Double/debiased machine learning: key ideas

Suppose we want to estimate the **partially linear regression (PLR)** model:

$$y_i = \tau D_i + g(\mathbf{x}_i) + u_i$$

where D_i is the "treatment" variable and $\mathbf{x}_i$ is a vector that contains p control variables and $g(\mathbf{x}_i)$ is an unknown function

We want to obtain an estimate of τ and make inferences about it

Suppose that

$$D_i = m(\mathbf{x}_i) + v_i, E(v_i|\mathbf{X}) = 0$$

where $m(\mathbf{x}_i)$ is an unknown function for the propensity score

Double/debiased machine learning: key ideas

In the classic case where both unknown functions are linear and $\mathbf{x}_i$ is known, there is usually no real problem

$\rightarrow$ We just regress y_i on D_i and $\mathbf{x}_i$, and the presence of the latter ensures that we obtain consistent estimates of τ. We don't even have to estimate the propensity score function

However, when we don't know $g(.)$, we run into the problem of **regularization bias**

Regularization bias

Any machine-learning estimator of $g(.)$ has to employ some sort of regularization, which induces bias

Let $\hat{g}_i$ denote an ML estimator for the $g(\mathbf{x}_i)$ part

One intuitive way to estimate τ is to regress $y_i - \hat{g}_i$ on D_i

If $\mathbf{y}$, $\mathbf{D}$, and $\mathbf{g}$ are vectors with typical elements y_i, D_i, and $g(\mathbf{x}_i)$, this yields the OLS estimator

$$\hat{\tau} = (\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'(\mathbf{y} - \hat{\boldsymbol{g}})$$

If $\hat{\boldsymbol{g}}$ were unbiased for $\mathbf{g}$, this estimator is unbiased

Unfortunately, for essentially all ML estimators, $\hat{\boldsymbol{g}}$ is not unbiased. This is especially true when the number of regressors in X is large

Regularization bias

Chernozhukov et al. (2018) shows that:

$$n^{1/2}(\hat{\tau} - \tau) = (n^{-1}\mathbf{D}'\mathbf{D})^{-1}n^{-1/2}\mathbf{D}'\mathbf{u} + (n^{-1}\mathbf{D}'\mathbf{D})^{-1}n^{-1/2}\mathbf{D}'(\mathbf{g} - \hat{\boldsymbol{g}})$$

while the first term has zero mean and variance that goes to zero at rate n

But the second term does not converge to zero fast enough

The quantity $n^{-1/2}\mathbf{D}'(\mathbf{g} - \hat{\boldsymbol{g}})$ is a sum of n terms, divided by $n^{1/2}$

Each of these terms has a non-zero mean, because $E(\mathbf{g} - \hat{\boldsymbol{g}}) \neq 0$

Although the bias diminishes as n increases, it always does so more slowly than $n^{-1/2}$

This does not imply that $\hat{\tau} - \tau$ diverges. But it converges more slowly than it should under standard asymptotics, and the bias can be large

How fast $\hat{\tau}$ converges to τ depends on how fast $\hat{\boldsymbol{g}}$ converges to $\mathbf{g}$. For all machine-learning methods, this is slower than $n^{-1/2}$

Double/debiased machine learning: key ideas

The way to overcome this regularization bias is to estimate two equations via machine learning

This is called **double machine learning**

The idea is to "partial out" the effects of $\mathbf{x}_i$ on D_i to obtain residuals $\hat{v}_i = D_i - \hat{m}(x_i)$, and on y_i to obtain $\hat{u}_i = y_i - \hat{g}_i$

Regressing $\hat{u}_i$ on $\hat{v}_i$ yields the double-ML estimator:

$$\hat{\tau} = [(\mathbf{D} - \hat{\boldsymbol{m}})'(\mathbf{D} - \hat{\boldsymbol{m}})]^{-1}(\mathbf{D} - \hat{\boldsymbol{m}})'(\mathbf{y} - \hat{\boldsymbol{g}})$$

This looks as if we are using the FWL theorem: we are regressing a vector of residuals on another vector of residuals

Chernozhukov et al. (2018) show that this estimator does not suffer of regularization bias

Amazingly, conventional standard errors for $\hat{\tau}$ are asymptotically valid

Sample splitting to remove bias induced by overfitting

Using sample splitting, i.e., estimate the nuisance models $\hat{g}(.)$ and $\hat{m}(.)$ on one part of the data (training data) and estimate τ on the other part of the data (test data), overcomes the bias induced by overfitting:

1. Randomly split the sample into halves: I and I^c
2. Use ML to predict Y from X on $I^c \rightarrow$ get the residuals $\mathbf{y} - \hat{\boldsymbol{g}}$ for observations in I
3. Use ML to predict D from X on $I^c \rightarrow$ get the residuals $\mathbf{D} - \hat{\boldsymbol{m}}$ for observations in I

4.Regress residuals on residuals: $\mathbf{y} - \hat{\boldsymbol{g}}$ on $\mathbf{D} - \hat{\boldsymbol{m}}$ to get the OLS estimate of τ for sample I: get $\hat{\tau}^{(1)}$

Sample splitting to remove bias induced by overfitting

5. Repeat Steps 2-4 but now using set I for prediction and set I^c for estimation: get $\hat{\tau}^{(2)}$ for sample I^c
6. The cross-fitted DML estimator is:

$$\tau = 0.5(\hat{\tau}^{(1)} + \hat{\tau}^{(2)})$$

In principle, you can make K splits of data... just like K-fold CV, it is **K-fold cross-fitting**

For the variance see the paper

Double/debiased machine learning: key ideas

Almost any machine-learning method can be used for the two first-stage regressions, that is, regressing y_i on $g(\mathbf{x}_i)$ and D_i on $m(\mathbf{x}_i)$

When number of regressors is large and sparsity seems plausible, one can use *lasso* estimates of $g(.)$ and $m(.)$

Whether or not the number of regressors is large, one can use regression trees, random forests or boosting

We can even use several different methods and then average their outputs, perhaps giving higher weights to ones that work better

Double/debiased machine learning: key ideas

DML approach can be generalized for the setting with heterogeneous treatment effects

Specifically, the Double ML literature considers 4 alternative models depending on whether:

- Treatment variable is **endogenous** (IV approach)
- Treatment effect (τ) is constant or heterogeneous across units

Partially linear regression model (PLR)

The **Partially linear regression model (PLR)** is the simples specification and takes the form:

$$y_i = \tau D_i + g(\mathbf{x}_i) + u_i, E(u_i|\mathbf{X}, D) = 0$$

where

$$D = m(x) + v, E(v_i|\mathbf{X}) = 0$$

Partially linear IV regression model (PLIV)

The **Partially linear IV regression model (PLIV)** model takes the form:

$$y_i = \tau D_i + g(x_i) + u_i, E(u_i|\mathbf{Z}, \mathbf{X}) = 0$$

$$z_i = m(\mathbf{x}_i) + v_i, E(v_i|\mathbf{X}) = 0$$

Interactive regression model (IRM)

The **Interactive regression (IRM)** model has been developed to handle models in which D_i enters **nonlinearly.** The model takes the form:

$$y_i = g(D_i, \mathbf{x}_i) + u_i, E(u_i|\mathbf{X}, \mathbf{D}) = 0$$

$$D_i = m(x_i) + v_i, E(v_i|\mathbf{X}) = 0$$

This model has been only developed for the case the treatment variable is binary, $D \in \{0, 1\}$. Under this specification the treatment effect is fully **heterogeneous**, and the target parameters of interest is the average treatment effect (ATE):

$$\tau = E[g(1, x) - g(0, x)]$$

Interactive IV model (IIVM)

If the treatment variable enters nonlinearly in the specification we have the **Interactive IV regression (IIVM)**. The models take the form:

$$y_i = g(D_i, \mathbf{x}_i) + u_i, E(u_i|\mathbf{Z}, \mathbf{X}) = 0$$

$$\mathbf{z}_i = m(\mathbf{x}_i) + v_i, E(v_i|\mathbf{X}) = 0$$

this case has been developed only for the case the treatment variable is binary, $D \in \{0, 1\}$ and there is a single instrument that is binary, $z \in \{0, 1\}$.