Lecture 11: Machine Learning

Generalized Random Forests

James Sears*
AFRE 891 SS24
Michigan State University

Table of Contents

Machine Learning for Causal Treatment Effect Estimation

Machine Learning for Causal Treatment

The goal of **Random Forests** is to estimate a **conditional mean**,

$$\mu(x_0) = E[Y \mid X = x_0]$$

But what if instead we wanted to estimate... **any** conditional function $\theta(x_0)$?

- Conditional linear regression coefficients
- Conditional causal treatment effects
- Conditional local average treatment effects using instruments
- Quantiles of the conditional distribution of $Y \mid X = x_0$
- Conditional class probabilities

Enter **Athey, Tibshirani, and Wager (2019)**.

Generalized Random Forests

Generalized Random Forests by Athey, Tibshirani, and Wager (2019) extends the ensemble methods of Random Forests to essentially any conditional function $\theta(x)$ that can be identified by local moment conditions.

They recast forests as an **adaptive locally weighted estimator**

1. Use the forest to calculate weighted set of neighbors for each observation (sound familiar?)
   - Employ problem-specific splitting rules to maximize the likelihood of capturing heterogeneity in $\theta(x)$
2. Use those neighbors to estimate $\theta(x)$ as the solution to a local moment equation

GRF Setup

More formally:

Suppose you have an IID sample with N observations.

You observe:

- O_i an observable quantity that encodes information relevant to estimating $\theta(\cdot)$
  - For nonparametric regression, this is just our our outcome $O_i = \{Y_i\}$
  - For treatment effect estimation, $O_i = \{Y_i, W_i\}$
  - Generally can contain more info
- Auxiliary covariates Xi

GRF

Goal: Estimate solutions to local estimating equations of the form

$$E[\psi_{\theta(x),\, \nu(x)}(O_i \mid X_i = x)] = 0$$

- $\theta(x)$ the parameter we care about (i.e. conditional mean, treatment effect, regression slope)

- $\nu(x)$ a (optional) nuisance parameter

--

- If conditional mean without noise, then $\psi_{\mu(x)}(Y_i) = Y_i - \mu(x)$

GRF Forest-Based Local Estimation

One approach to estimating functions like $\theta(x)$:

- Find $\hat{\theta}(x), \hat{\nu}(x)$ that solve

$$\sum_{i=1}^{n} \alpha_i \psi_{\hat{\theta}(x),\hat{\nu}(x)}(O_i) = 0$$

- α_i: weights measuring relevance of the i^{th} observation for fitting $\theta(\cdot)$ at x
  - Traditionally obtained using a kernel function (and sometimes a bandwidth)
  - Traditionally sensitive to curse of dimensionality

GRF Forest-Based Local Estimation

GRF Approach: use forests to derive α_i.

1. Grow $b = 1, \ldots, B$ trees
2. Choose weights $\alpha_i(x)$ to capture how frequently the i^{th} training example falls into the same leaf as x.

Weights within a given tree, $\alpha_{bi}(x)$, are calculated as

$$\alpha_{bi}(x) = \frac{I[X_i \in L_b(x)]}{|L_b(x)|}$$

- $L_b(x)$ the set of observations falling within the same leaf as x

Overall weights for observation i are then the average across all trees:

$$\alpha_i(x) = \frac{1}{B} \sum_{b=1}^{B} \alpha_{bi}(x)$$

GRF Weights

For example, we can reframe random forests and our estimate of the conditional mean function as

$$\hat{\mu}(x) = \sum_{i=1}^{n} \frac{1}{B} \sum_{b=1}^{B} Y_i \frac{I[X_i \in L_b(x)]}{|L_b(x)|}$$

Or as the weighted sum of single tree predictions $\hat{\mu}_b(\cdot)$:

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\mu}_b(x)$$

Or more simply, as

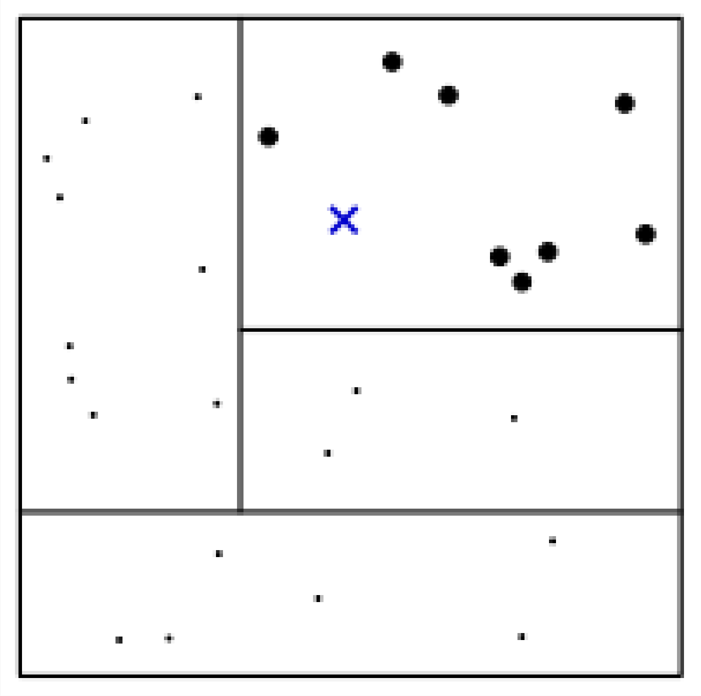
$$\hat{\mu}(x) = \sum_{i=1}^{n} Y_i \alpha_i(x)$$

GRF Weights Visually

Let's see how these weights are developed with a simple example: a forest with three trees and two covariates.

- x the point of interest
  - lines our splits
  - darker dots the points within the same leaf

Tree 1

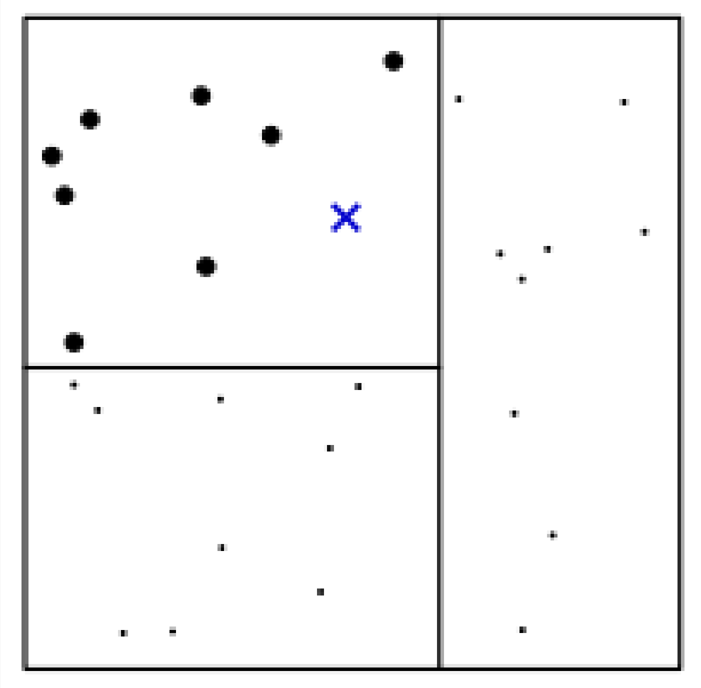


GRF Weights Visually

Let's see how these weights are developed with a simple example: a forest with three trees and two covariates.

- x the point of interest
  - lines our splits
  - darker dots the points within the same leaf

Tree 2

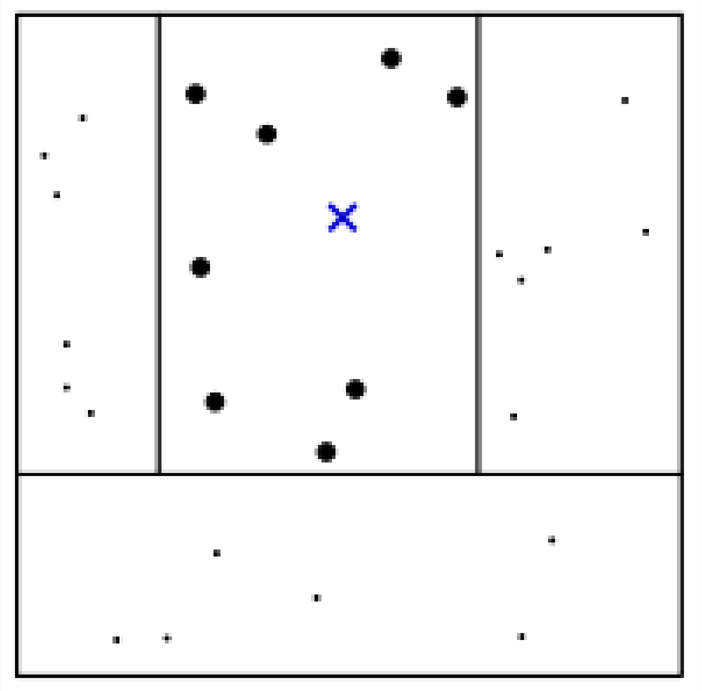


GRF Weights Visually

Let's see how these weights are developed with a simple example: a forest with three trees and two covariates.

- x the point of interest
  - lines our splits
  - darker dots the points within the same leaf

Tree 3

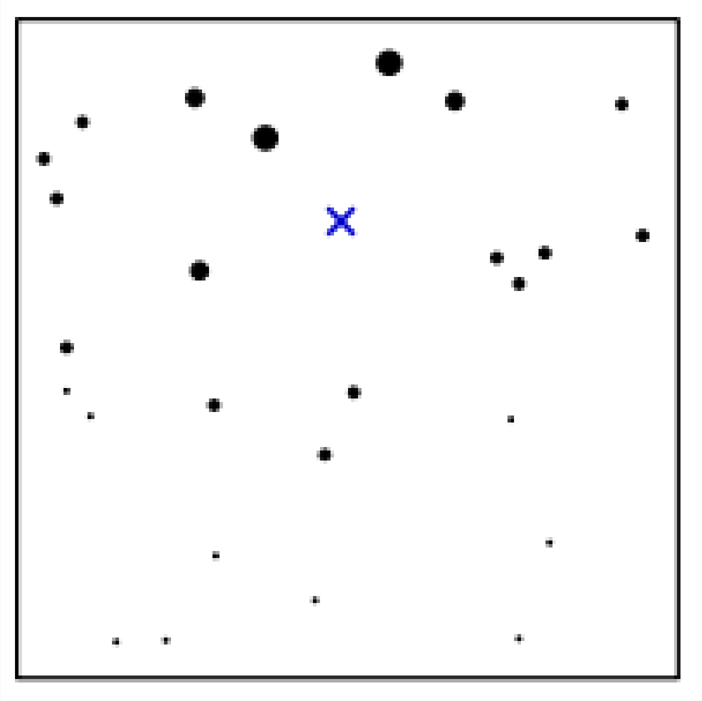


GRF Weights Visually

Let's see how these weights are developed with a simple example: a forest with three trees and two covariates.

Then we average across all the individual tree-based weights to obtain the overall $\alpha_i(x)$.

Forest Weights for x



Honesty

One additional benefit of using GRF: growing **honest** forests

Honesty aims to reduce bias in tree predictions by using **distinct subsamples** for 1. Building the tree, and 2. Making predictions

Classic Random Forest

- Draws a single subsample, use that subsample for both choosing a tree's splits and making predictions in that tree

Honest Forests

- Draw a training sample
- Use part of that sample to choose the tree's splits
- Use the rest of the training sample to make predictions
- Prune away any remaining empty leaves

GRF Functions

Conveniently for us, the **grf** package contains tons of features and **extensive documentation** to estimate a wide range of forests, including...

Causal Treatment Effects

Approach	Forest Function
Causal Treatment Effects	`causal_forest()`
Causal Treatment Effects with right-censored outcomes	`causal_survival_forest()`
Multi-arm/multi-outcome causal treatment effects	`multi_arm_causal_forest()`

GRF Functions

Conveniently for us, the **grf** package contains tons of features and **extensive documentation** to estimate a wide range of forests, including...

Moments of Conditional Distributions

Approach	Forest Function
Conditional Mean	`regression_forest()`
Multi-outcome conditional means	`multi_regression_forest()`
Right-censored Survival	`survival_forest()`
Conditional Quantiles	`quantile_forest()`
Conditional Class Probabilities	`probability_forest()`

GRF Functions

Conveniently for us, the **grf** package contains tons of features and **extensive documentation** to estimate a wide range of forests, including...

Regression Outcomes

Approach	Forest Function
Conditional Linear Model	`lm_forest()`
Local Average Treatment Effects (IV)	`instrumental_forest()`

Causal Forests

When $W_i \in \{0,1\}$ and **unconfoundedness holds**, we can estimate heterogeneous causal treatment effects with **causal forests**.

1. Build Phase: greedily choose splits to maximize the **squared difference in subgroup treatment effects**

$$n_L n_R (\hat{\tau}_L - \hat{\tau})_R)^2$$

- $\hat{\tau}$'s chosen through a centered residual-on-residual regression (Robinson 1988)

Causal Forests

2. Estimate $\tau(x)$, where

$$\tau(x) := lm\ \left(Y_i - \hat{m}^{-1}(X_i) \sim W_i - \hat{e}^{-1}(X_i),\ \textbf{weights} = \alpha_i(x)\right)$$

- $\left(Y_i - \hat{m}^{-1}(X_i),\ W_i - \hat{e}^{-1}(X_i)\right)$: orthogonalized outcome/treatment
  - Residual after "regressing out" the main effect of X_i on Y_i (W_i)
  - $\hat{m}^{-1}(X_i)$, $\hat{e}^{-1}(X_i)$ obtained from separate regression forests

Causal Forests Example

Let's load in some data to see how we can run causal forests, using a sample from the **National Study of Learning Mindsets**.

ARTICLE

OPEN

https://doi.org/10.1038/s41586-019-1466-y

A national experiment reveals where a growth mindset improves achievement

David S. Yeager[1*], Paul Hanselman[2*], Gregory M. Walton[3], Jared S. Murray[1], Robert Crosnoe[1], Chandra Muller[1], Elizabeth Tipton[4], Barbara Schneider[5], Chris S. Hulleman[6], Cintia P. Hinojosa[7], David Paunesku[8], Carissa Romero[9], Kate Flint[10], Alice Roberts[10], Jill Trott[10], Ronaldo Iachan[10], Jenny Buontempo[1], Sophia Man Yang[1], Carlos M. Carvalho[1], P. Richard Hahn[11], Maithreyi Gopalan[12], Pratik Mhatre[1], Ronald Ferguson[13], Angela L. Duckworth[14] & Carol S. Dweck[3]

```r
nslm <- read.csv("data/NSLM.csv")
```

Causal Forests Example

Taking a look at the data, we have the following variables:

- `schoolid` the ID of randomly selected US public high schools
- `Y`: continuous measure of achievement
- `Z`: treatment status
- `S3`: students' self-reported expectations for future success
- `C1`: student race (categorical)
- `C2`: student gender (categorical)
- `C3`: first-generation status (categorical)
- `XC`: school urbanicity (categorical)
- `X1`: school-level mean of students' fixed mindsets
- `X2`: school achievement level (test scores + college prep)
- `X3`: school racial minority composition (% black, latino, native american)
- `X4`: school poverty (% in familes below FPL)
- `X5`: school size

Causal Forests Example

We want to answer the following research questions:

1. Was a nudge-like mindset intervention designed to instill a "growth mindset" in students effective at improving achievement?

2. Do schools' prior achievement levels (`X2`) and pre-existing mindset norms (`X1`) effect the magnitude of this effect?

Let's find out!

Causal Forests Example

We *could* just run an (interacted) OLS regression...

```
reg1 ← feols(Y ~ Z + S3 + C1 + C2 + C3 + XC + X1 + X2 + X3 + X4 + X5, data  = nsl
reg2 ← feols(Y ~ Z + Z:X2 + Z:X3 + S3 + C1 + C2 + C3 + XC + X1 + X2 + X3 + X4 + X
etable(reg1, reg2, keep = c("Z", "X2", "X1"))
```

	reg1	**reg2**
Dependent Var.:	Y	Y
Z	0.2549*** (0.0115)	0.2539*** (0.0115)
X1	-0.0954*** (0.0073)	-0.0954*** (0.0073)
X2	-0.0163. (0.0087)	-0.0263** (0.0098)
Z x X2		0.0313* (0.0141)
Z x X3		0.0152 (0.0134)

Causal Forests Example

...which gives us an estimate of the mean treatment effect and mean mediating effect of `X1/X2`.

Instead, let's use `causal_forest()` to obtain heterogeneous treatment effects for each school.

Let's first build some data objects we'll use as arguments.

1. Convert `C1` and `XC` to dummies:

- `C1` has 15 levels, so rather than code each one by hand let's use `model.matrix`
  - Argument: formula

```
# expand C1 categorical variable into dummies (with intercept)
C1_exp ← model.matrix(~ factor(nslm$C1) + 0)
```

Causal Forests Example

Let's first build some data objects we'll use as arguments.

Repeat for `XC`:

```
XC_exp <- model.matrix(~ factor(nslm$XC) + 0)
```

Causal Forests Example

Next, build the predictor matrix

```r
# first select all covariates except for XC and C1
X ← select(nslm, -schoolid:-Y, -XC, - C1)

# Add in the dummies for XC and C1
Xmat ← cbind(X, C1_exp, XC_exp)
```

And get the outcome and treatment vectors along with the school clusters:

```r
Y ← nslm$Y # our outcome
Z ← nslm$Z # our treatment indicator
schoolid ← nslm$schoolid
```

Causal Forests Example

To perform the residual-on-residual regression step, our forest wants predictions for $\hat{Y}$ and $\hat{W}$. We can either

1. Omit them from the forest setup
    - The forest will estimate them for us in a separate regression forest
2. Run the initial regression forests manually and supply the predictions

Let's do the latter, but first an important consideration...

Causal Forests Example

As with random forests, by default GRF methods assume **independent observations**.

Here, we know that treatment assignment occurred at the **school level**, so students' treatment is dependent on which school that they're at.

The good news is, we can account for this **clustering** within our forest to draw our random units at the school level.

- `clusters` the clustering variable (here our `schoolid` factor)
- `equalize.cluster.weights = TRUE` ensures we draw the same fraction frmo each cluster

Causal Forests Example

Running the initial regression forests for our $\hat{Y}$ and $\hat{W}$ predictions:

```
set.seed(1)
Y_forest ← regression_forest(X = Xmat, Y = Y,
                              clusters = schoolid,
                              equalize.cluster.weights = TRUE)
Y_hat ← predict(Y_forest)$predictions

W_forest ← regression_forest(X = Xmat, Y = Z, clusters = schoolid, equal:
W_hat ← predict(W_forest)$predictions
```

Causal Forests Example

Setting up the causal forest:

- `X`: the covariate matrix

```
# Define the random forest
cf ← causal_forest(X = Xmat,
                    Y = Y,
                    Y.hat = Y_hat,
                    W = Z,
                    W.hat = W_hat,
                    clusters = schoolid,
                    equalize.cluster.weights =
                    num.trees = 200,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Causal Forests Example

Setting up the causal forest:

- `X`: the covariate matrix
- `Y`: the outcome vector (our measure of student achievement)
- `W`: the treatment vector

```r
# Define the random forest
cf <- causal_forest(X = Xmat,
                    Y = Y,
                    W = Z,
                    Y.hat = Y_hat,
                    W.hat = W_hat,
                    clusters = nslm$schoolid,
                    equalize.cluster.weights =
                    num.trees = 2000,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Causal Forests Example

Setting up the causal forest:

- `X`: the covariate matrix
- `Y`: the outcome vector (our measure of student achievement)
- `W`: the treatment vector
- `Y.hat`: our prediction of Y as a function of X
- `W.hat`: our prediction of W as a function of X

```r
# Define the random forest
cf <- causal_forest(X = Xmat,
                    Y = Y,
                    W = Z,
                    Y.hat = Y_hat,
                    W.hat = W_hat,
                    clusters = nslm$schoolid,
                    equalize.cluster.weights =
                    num.trees = 200,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Causal Forests Example

Setting up the causal forest:

- `clusters`: the cluster identifier vector
- `equalize.cluster.weights`: whether to draw equal sample sizes from each cluster

```r
# Define the random forest
cf <- causal_forest(X = Xmat,
                    Y = Y,
                    W = Z,
                    Y.hat = Y_hat,
                    W.hat = W_hat,
                    clusters = nslm$schoolid,
                    equalize.cluster.weights =
                    num.trees = 200,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Causal Forests Example

Setting up the causal forest:

- `clusters`: the cluster identifier vector
- `equalize.cluster.weights`: whether to draw equal sample sizes from each cluster
- `num.trees`: size of forest to grow (default is 2000)

```r
# Define the random forest
cf ← causal_forest(X = Xmat,
                   Y = Y,
                   W = Z,
                   Y.hat = Y_hat,
                   W.hat = W_hat,
                   clusters = nslm$schoolid,
                   equalize.cluster.weights =
                   num.trees = 200,
                   honesty = TRUE,
                   honesty.fraction = 0.5,
                   tune.parameters = "all"
                   )
```

Causal Forests Example

Setting up the causal forest:

- `honesty`: whether or not to grow honestly (default to TRUE)
- `honesty.fraction`: the subsample portion used for growing trees (defaults to 50%)

```
# Define the random forest
cf ← causal_forest(X = Xmat,
                   Y = Y,
                   W = Z,
                   Y.hat = Y_hat,
                   W.hat = W_hat,
                   clusters = nslm$schoolid,
                   equalize.cluster.weights =
                   num.trees = 200,
                   honesty = TRUE,
                   honesty.fraction = 0.5,
                   tune.parameters = "all"
                   )
```

Causal Forests Example

Setting up the causal forest:

- `tune.parameters`: which/whether to tune parameters
  - `none, all, sample.fraction, mtry, min.node.size, honesty.fraction, honesty.prune.leaves, alpha, imbalance.penalty`

```r
# Define the random forest
cf <- causal_forest(X = Xmat,
                    Y = Y,
                    W = Z,
                    Y.hat = Y_hat,
                    W.hat = W_hat,
                    clusters = nslm$schoolid,
                    equalize.cluster.weights =
                    num.trees = 200,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Variable Importance

Looking at **variable importance** with `variable_importance(forest)`:

- Scaled count of how frequently variables were chosen to split on (i.e. that they maximized treatment effect heterogeneity)

covar	**imp**
X1	0.214
X2	0.149
X3	0.137
X4	0.131
X5	0.0915
S3	0.0656

Split Frequencies

Alternatively, we can obtain the raw split frequencies with…

`split_frequencies(forest, max.depth)`

```
# getting split frequencies 4 branches deep
splitfreq <- split_frequencies(cf, max.depth = 4)
colnames(splitfreq) <- colnames(Xmat)
rownames(splitfreq) <- paste("Depth", 1:4)
splitfreq[,1:8]
```

```
##          S3 C2 C3 X1 X2 X3 X4 X5
## Depth 1   9  9  5 50 33 28 29 15
## Depth 2  38 23 19 57 45 59 38 59
## Depth 3  90 52 44 82 77 75 68 69
## Depth 4 112 74 65 70 64 63 70 80
```

Variable Selection

Our variable importance and split frequencies reveals useful information about the covariates: some **really don't matter**

Since we're only considering a random subset of covariates at each potential split, by including covariates without explanatory power for treatment effect heterogeneity, we're likely **adding noise** and **diluting** the performance of our forest.

Solution: re-run the tree, keeping only the **most important variables**

- Athey and Wager (2019): keep only variables greater than mean importance

Variable Selection

Selecting the "important" covariates:

```
varimp <- variable_importance(cf)
X_sel <- Xmat[,which(varimp> mean(varimp))]
colnames(X_sel)
```

```
## [1] "S3"                 "C2"                 "X1"                 "X2"
## [5] "X3"                 "X4"                 "X5"                 "factor(nslm$C1
```

Causal Forests Example

And rerunning the forest with just these variables:

```
cf_sel ← causal_forest(X = X_sel,
                    Y = Y,
                    W = Z,
                    num.trees = 2000,
                    honesty = TRUE,
                    honesty.fraction = 0.5,
                    tune.parameters = "all"
                    )
```

Causal Forests Treatment Effects

Use the `average_treatment_effect()` function to get one of several **average treatment effects**:

- `target.sample = "all"`: The ATE
  - $E[Y(1) - Y(0)]$
- `target.sample = "treated"`: The ATT
  - $E[Y(1) - Y(0) \mid W_i = 1]$

Causal Forests Treatment Effects

Use the `average_treatment_effect()` function to get one of several **average treatment effects**:

- `target.sample = "control"`: The average treatment effect on the control
  - $E[Y(1) - Y(0) \mid W_i = 0]$
- `target.sample = "overlap"`: The overlap-weighted average treatment effect on the control
  - $E[e(X)(1 - e(X))(Y(1) - Y(0))]/E[e(X)(1 - e(X))]$
  - Where $e(X) = P[W_i = 1 \mid X_i = x]$

Causal Forests Treatment Effects

Use the `average_treatment_effect()` function to get one of several **average treatment effects**:

```
ate <- average_treatment_effect(cf_sel, target.sample = "all")
ate
```

```
##    estimate     std.err
## 0.25446427 0.01082919
```

Causal Forests Treatment Effects

Retrieving the out-of-bag predictions for $\tau(X)$ with `predict()`:

```
tauhat_oob ← predict(cf_sel, estimate.variance = TRUE)
head(tauhat_oob)
```

predictions	variance.estimates	debiased.error	excess.error
0.238	0.00167	0.0151	1.13e-05
0.196	0.00194	0.00903	1.52e-05
0.241	0.00155	0.0344	1.12e-05
0.245	0.00196	0.161	1.16e-05
0.226	0.00213	0.355	1.15e-05
0.242	0.00249	0.0934	1.07e-05

Causal Forest Error

In addition to our predicted treatment effects and variance estimate, we get **two types of error**:

Debiased Error

- Estimate of the **"R-loss" Criterion**
  - i.e. the **predictive fit** of our forest

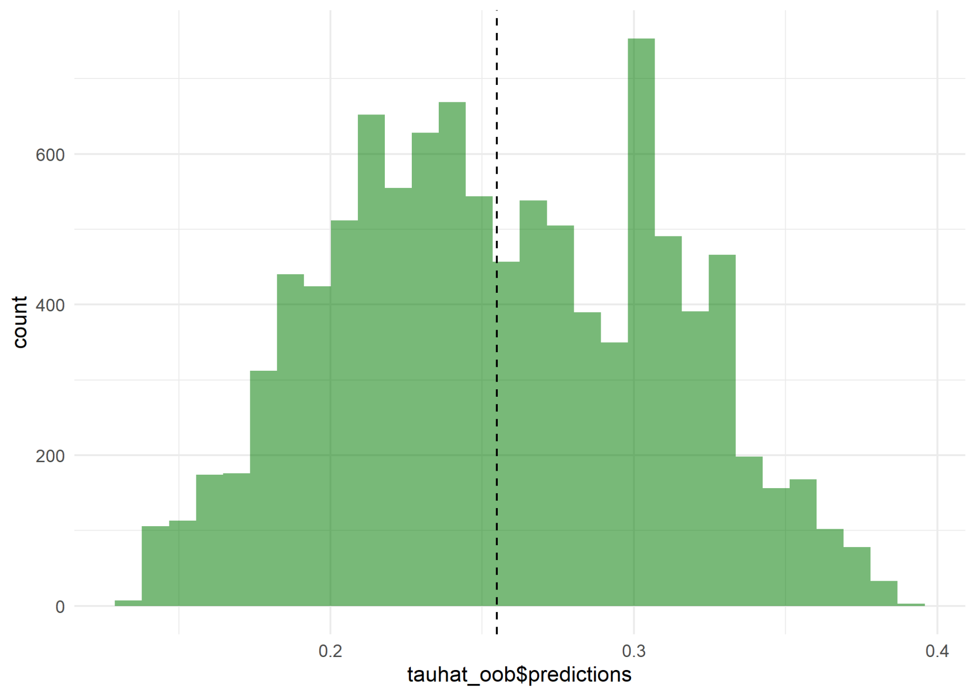
Excess Error

- Error resulting from **random nature of forests**
  - i.e. how **unstable** our estimates would be if we re-grew forests with the same data/setup
  - We want this to be very small relative to estimate's variance

Treatment Effects

Plotting the distribution of individual treatment effects (and relative to the OLS mean estimate:)

```
ggplot() +
  geom_histogram(aes(tauhat_oob$predictions), fill = "forestgreen", alpha = 0.6) +
  geom_vline(aes(xintercept = reg1$coefficients["Z"]), linetype = "dashed") +
  theme_minimal()
```



ATE Subset Comparison

We can also use `subset` to see differences in top/bottom treatment effects

```
# Getting boolean vectors of top/bottom quartile treatment effects
high_effect <- tauhat_oob$predictions > quantile(tauhat_oob$predictions, (
low_effect <- tauhat_oob$predictions < quantile(tauhat_oob$predictions, 0
```

ATE Subset Comparison

We can also use `subset` to see differences in top/bottom treatment effects

```
# getting ATE for each group
ate_high = average_treatment_effec
ate_high
```

```
# getting ATE for each group
ate_low = average_treatment_effect
ate_low
```

```
##    estimate    std.err
## 0.35006924 0.05311834
```

```
##    estimate    std.err
## 0.23268740 0.03414574
```

```
## [1] "95% CI for difference in ATE: 0.117 +/- 0.124"
```

Causal Forests Example

Re-estimating the forest **without clusters** shows the importance of accounting for the cluster assignment:

```
cf_noclust <- causal_forest(X = X_sel,
                           Y = Y,
                           Y.hat = predict(Y_forest)$predictions,
                           W = Z,
                           W.hat = predict(W_forest)$predictions,
                           num.trees = 2000,
                           honesty = TRUE,
                           honesty.fraction = 0.5,
                           tune.parameters = "all"
                           )
```

Test Calibration

Looking at the **Test Calibration** results for each forest shows the importance of clustering.

`test_calibration()` Regresses the forest error for prediction of Y on **two terms:**

1. `Mean Prediction`: indication of how well the forest captures the mean ATE
   - Coefficient indistinguishable from 1 $\longrightarrow$ accurate mean prediction
2. `Differential Prediction:` how well the forest captures heterogeneity in the ATE
   - Coefficient distinguishable from 0 $\longrightarrow$ confirms existence of heterogeneous treatment effects
   - Coefficient indistinguishable from 1 $\longrightarrow$ accurately capture heterogeneity

Test Calibration

First for the non-clustered forest:

```
test_calibration(cf_sel)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value    Pr(>t)
## mean.forest.prediction         0.997121   0.042448 23.4903 < 2.2e-16 ***
## differential.forest.prediction 1.148856   0.202310  5.6787 6.969e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test Calibration

Looking at the **Test Calibration** results for each forest shows the importance of clustering.

Next for the unclustered forest:

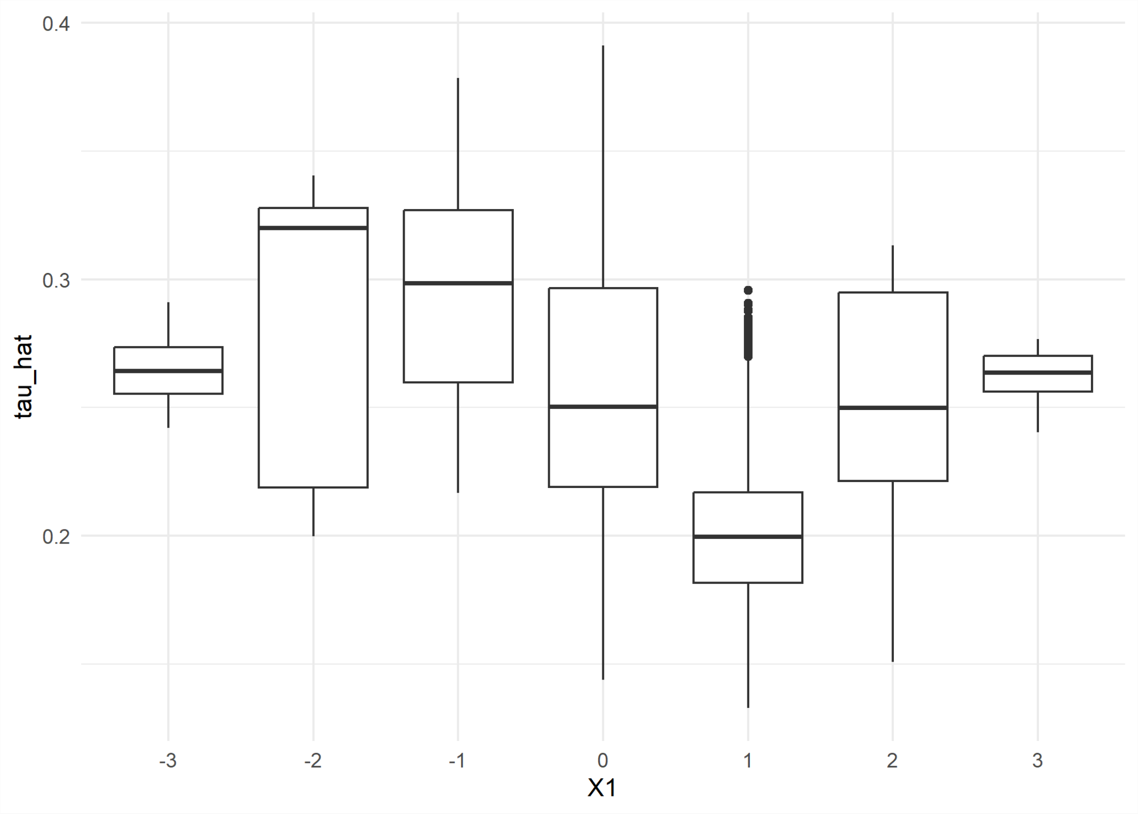
```
test_calibration(cf_noclust)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value    Pr(>t)
## mean.forest.prediction         1.015750   0.045235 22.4549 < 2.2e-16 ***
## differential.forest.prediction 0.531436   0.122606  4.3345 7.374e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Treatment Effect Heterogeneity

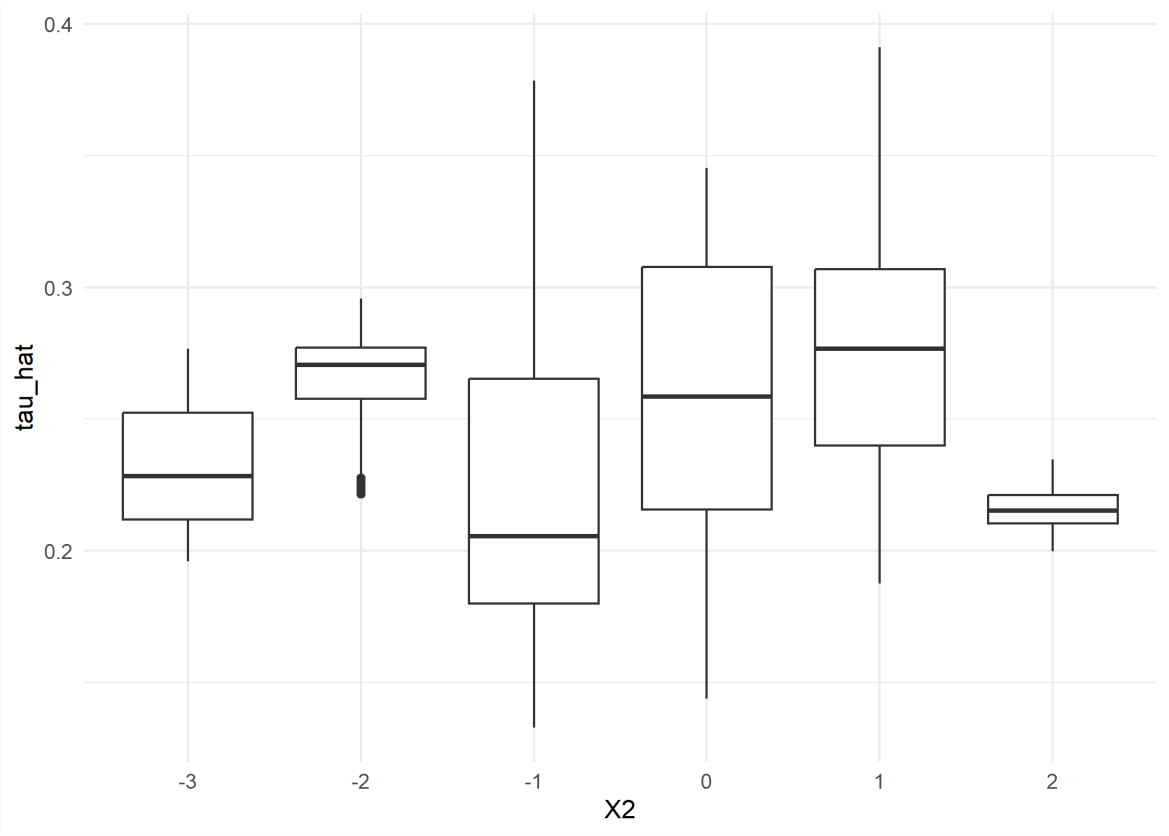
It looks like there's a clear effect of the intervention nudge on achievement, but what about the mediating effects of `X1` and `X2`?

First, let's look at the variation in treatment effects by the values of `X1`:



Treatment Effect Heterogeneity

Repeating for pre-existing achievement levels (`X2`):



It appears that the nudge had the least impact in schools with the highest pre-existing achievement level (`X2`)

Covariate Correlation

One potential limitation: there are a lot of factors at the school-level associated with pre-existing achievement levels.

```
x2_reg <- feols(X2 ~ X1 + X3 + X4 + X5 + C2 + C3  | XC + C1, data = nslm)
etable(x2_reg)
```

	x2_reg
Dependent Var.:	X2
X1	-0.2858 (0.1592)
X3	-0.2635 (0.1368)
X4	-0.0902 (0.0859)
X5	0.2635. (0.1117)
C2	0.0023 (0.0155)

Simulations

Solution: use a new "simulated" covariate matrix to predict treatment effects if schools had different achievement levels, holding other characteristics fixed.

Let's write a function that will let us obtain treatment effect predictions if we swap the value of `X2` with a particular value in all rows:

```r
sim_x2 <- function(val){
  # replace all X2 values in the Xmat with the specified value
  X <- mutate(Xmat, X2 = val)

  # Estimate forest
  cf_sim <- causal_forest(X = X_sel,
                          Y = Y,
                          W = Z,
                          num.trees = 200, # fewer trees for expediency of estimation
                          honesty = TRUE,
                          honesty.fraction = 0.5,
                          tune.parameters = "all"
                          )

  res <- data.frame(preds = predict(cf_sim)$predictions) %>%
    mutate(X2_val = val)
  return(res)
}
```

Simulations

Next, build a vector of X2 values to predict for:

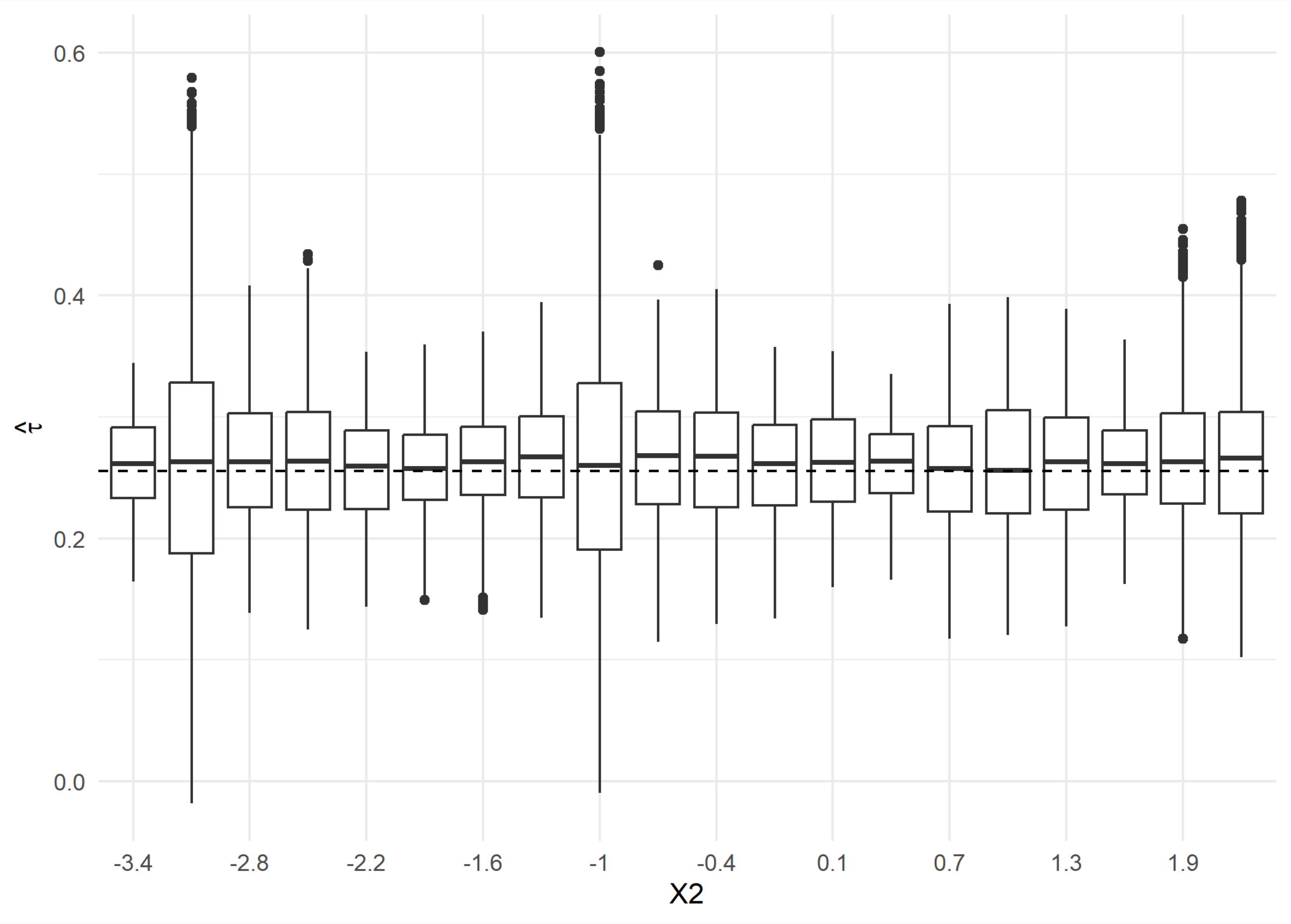
```
# get range of X2 to predict over
min_x2 <- min(Xmat$X2)%>% round(2)
max_x2 <- max(Xmat$X2) %>% round(2)
x2_seq <- seq(min_x2, max_x2, length.out = 20)
```

And running:

```
# Vectorizing and saving out
sim_x2 <- map_dfr(x2_seq, sim_x2)
saveRDS(sim_x2, "output/x2_sim_df.rds")
```

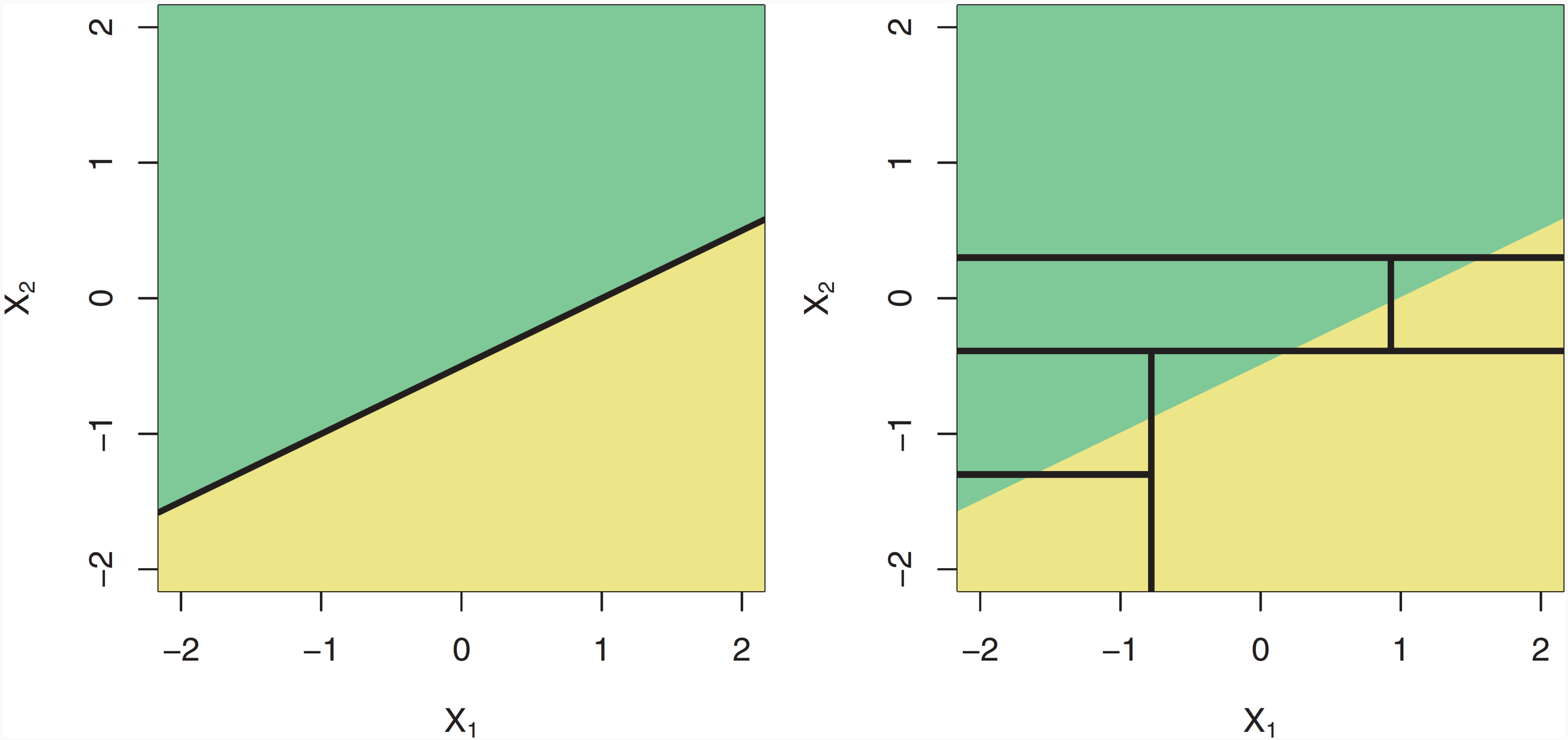
Simulations

Plotting the simulation results:



Linear Boundaries

What if I want to use these methods to estimate heterogeneous treatment effects but I'm concerned about replicating **linear boundaries**?



A: local linear forests

Local Linear Forests

[Developed in Friedberg et al. (2021)], local linear forests combine the local smoothness benefits of local linear regression with the high-dimensional flexibility of random forests.

Random Forests

- Highly data-adaptive
- Struggle to capture smoothness in conditional mean function

Local Linear Regression

- Great at capturing locally smooth signals
- Quickly deteriorates due to curse of dimensionality
  - Relies on Euclidean distance, which loses locality even in 4-5 dimensions

Local Linear Forests

Local linear forests function as a Two-stage adaptive kernel method:

1. Grow a random forest with splits chosen to minimize the sum of squared errors between the resulting two branches
    - Uses a standard splitting procedure but splitting on the **residuals from a ridge regression**
    - Partitions the covariate space based on local effects, captures existing linear signals
2. Use the forest to obtain weights $\alpha_i(x_0)$
3. Rather than use the weights to fit the local average at x_0 as in random forests, Use the $\alpha_i(x_0)$ weights to fit a **local linear regression**
    - Includes a ridge penalty for regularization

Local Linear Forests

More formally:

Consider two parameters:

1. $\mu(x_0)$: the local average at x_0
2. $\theta(x_0)$: the slope of the local line
   - Correction for local trend in $X_i - x$

Goal: solve for estimates $\hat{\mu}(x_0)$, $\hat{\theta}(x_0)$ that minimize

$$\sum_{i=1}^{n} \underbrace{\alpha_i(x_0)}_{\text{Weights}}(Y_i - \mu(x_0) - \underbrace{(X_i - x_0)\theta(x_0)}_{\text{Local Trend Correction}})^2 + \underbrace{\lambda||\theta(x_0)||_2^2}_{RidgePenalty}$$

Local Linear Forests

Local linear forests also extend to causal forests:

1. Use local linear forests to obtain the estimates for $\left(\hat{m}^{-1}(X_i),\ \hat{e}^{-1}(X_i))\right)$
    - i.e. the leave-one-out initial predictions for Y and W obtained from auxiliary forests

Local Linear Forests

Local linear forests also extend to causal forests:

2. Estimate conditional average treatment effect $\hat{\tau}$ as the arg min of

$$\sum_{i=1}^{n} \alpha_i(x_0)\Big(Y_i - \hat{m}^{-1}(x_i) - a - (X_i - x_0)\theta_a \\ - (\tau + \theta_t(X_i - x_0))(W_i - \hat{e}^{-1}(X_i))\Big)^2 \\ + \lambda_\tau ||\theta_\tau||_2^2 + \lambda_a ||\theta_a||_2^2$$

- a intercept (0 if $\Big(\hat{m}^{-1}(X_i),\ \hat{e}^{-1}(X_i)\Big)$ estimates are accurate)
- λ_τ and λ_a: ridge penalties for local trends in treatment and outcome
  - As these get large, we recover a standard causal forest

Local Linear Forests

To estimate a local linear causal forest:

1. Use `ll_regression_forest()` instead of `regression_forest()` to obtain the initial estimates of $\left(\hat{m}_{ll}^{-1}(X_i),\ \hat{e}_{ll}^{-1}(X_i))\right)$

```
# Got LL forest predictions of Y
Y_ll_forest ← ll_regression_forest(X = Xmat, Y = Y,
                                   clusters = schoolid,
                                   equalize.cluster.weights = TRUE,
                                   # make forest splits on ridge residuals
                                   enable.ll.split = TRUE)
Y_ll_hat ← predict(Y_ll_forest,
                   # standardize ridge penalty by covariance
                   ll.weight.penalty = T)$predictions
# Repeat for W
W_ll_forest ← ll_regression_forest(X = Xmat, Y = Z, clusters = schoolid,
W_ll_hat ← predict(W_ll_forest, ll.weight.penalty = T)$predictions
```

Local Linear Forests

To estimate a local linear causal forest:

2. Run Causal forest using the local linear forest estimates from 1.

```
cf_sel_ll ← causal_forest(
  # unchanged parameters:
  X = X_sel, Y = Y, W = Z, clusters = nslm$schoolid, equalize.cluster.wei
  # changed inputs:
                    Y.hat = Y_ll_hat,
                    W.hat = W_ll_hat,
                    )
```

Local Linear Forests

To estimate a local linear causal forest:

3. Use local linear arguments of `predict()` when generating predictions for τ

```
tauhat_ll_oob <- predict(cf_sel_ll,
                         linear.correction.variables = 1:ncol(X_sel),
                         ll.weight.penalty = TRUE
                         )
```

Local Linear Forests

Comparing overall fit (Root MSE) between the two:

```
rmse_cf <- sqrt(sum((cf_sel$Y.orig - cf_sel$Y.hat)^2)/length(cf_sel$Y.orig
rmse_ll <- sqrt(sum((cf_sel_ll$Y.orig - cf_sel_ll$Y.hat)^2)/length(cf_sel_
data.frame(Forest = c("Causal Forest", "Local Linear Causal Forest"),
           RMSE = c(rmse_cf, rmse_ll) %>% round(4))
```

Forest	RMSE
Causal Forest	0.533
Local Linear Causal Forest	0.551

Local Linear Forests

Comparing test calibration results:

```
test_calibration(cf_sel)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value    Pr(>t)
## mean.forest.prediction         0.997121   0.042448 23.4903 < 2.2e-16 ***
## differential.forest.prediction 1.148856   0.202310  5.6787 6.969e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Local Linear Forests

Comparing test calibration results:

```
test_calibration(cf_sel_ll)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value Pr(>t)
## mean.forest.prediction         0.963609   0.072971 13.2053 <2e-16 ***
## differential.forest.prediction -9.150270   2.981415 -3.0691 0.9989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Local Linear Forests

And comparing predictions from the two approaches:

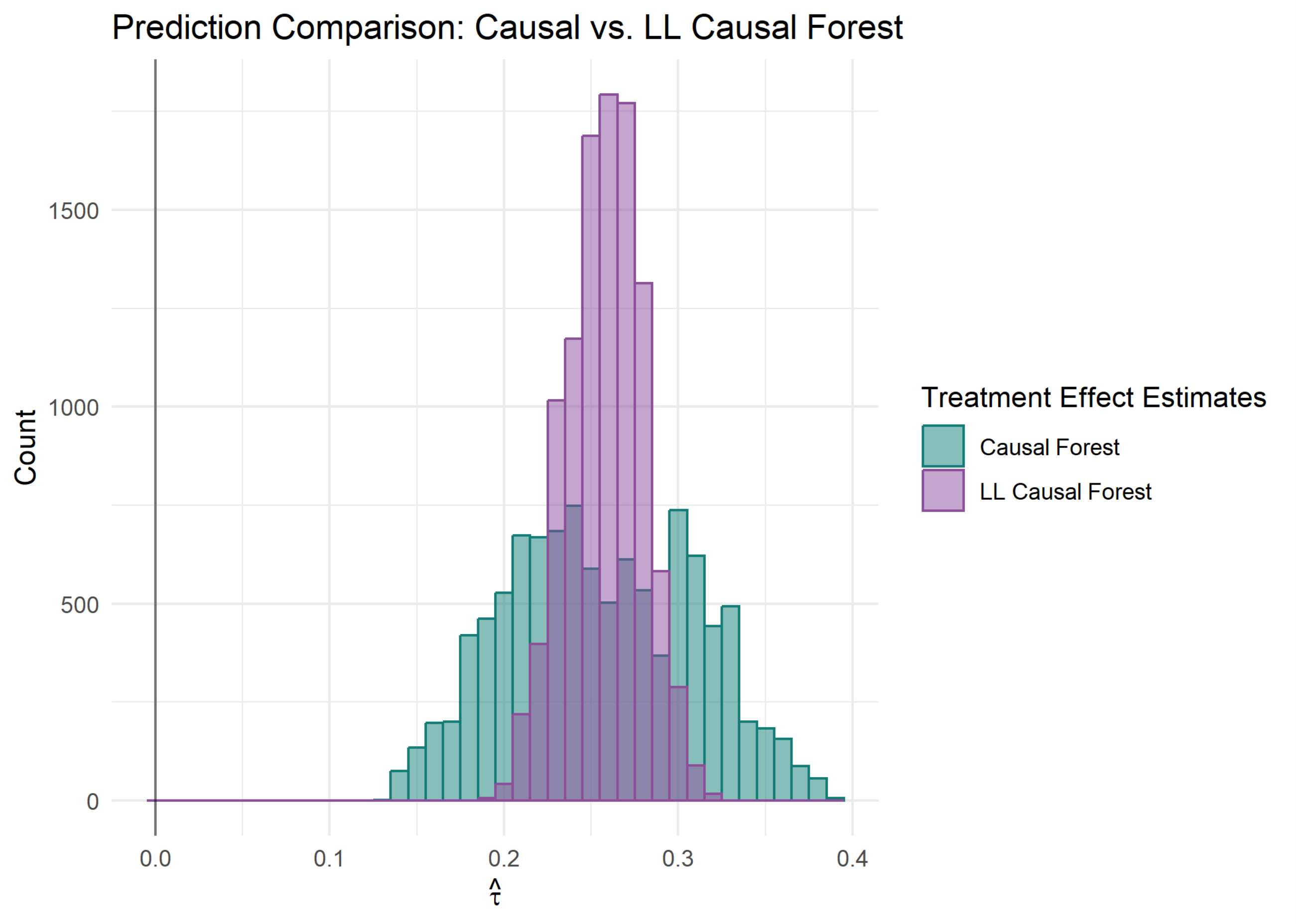


Table of Contents