Causal Forest

Alessandro Caggia

June 2025

Abstract

1. Introduction

1.1. The Big Picture

- Our goal in this lecture is to estimate heterogeneous treatment effects based on X.
- We would like to recover the full distribution of the ATE(x):

$$\mathbb{E}(Y_{i1} - Y_{i0} \mid X_i = x)$$

- Traditional Heterogeneity Analysis focuses on a few covariates, ideally pre-specified based on theoretical grounds. It just runs OLS with interactions or splits the samples in subgroups (as done up to now).
- Machine Learning tools provide more flexible non-parametric estimators that allow us to find the most important sources of heterogeneity based on **ALL** available baseline covariates (and all their interactions).
  - *No theory, no econ intuition. Just a plan to explore the data. Some form of structured data mining.*
- We can detect unexpected sources of heterogeneity without the risk of p-hacking or data mining.

A causal forest allows us to estimate heterogeneous causal eects without any restriction on the number of covariates. The method can be used to explore any previously conducted RCT in order to discover subpopulations with high or low treatment eects (+ CIs)

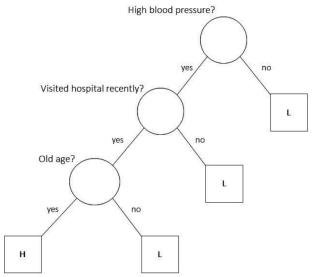
1.2. Basic Concepts

- We want to select covariates that maximize **treatment heterogeneity**.
- What we have ahead:
  - Classification and Regression Trees (CART)
  - Random Forests
  - Causal Trees
  - Causal Forests

2. Classification and Regression Trees (CART)

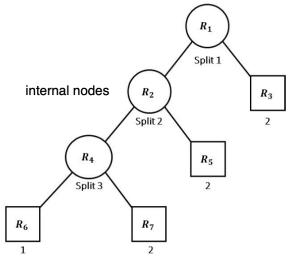
2.1. Classification Trees

- **General classification problem: Notation:**
  - Let $x = (x_1, x_2, \dots) \in \mathcal{X}$ be a vector of measurements (features).
  - Assume J different possible classes.
- **Definition: Classifier**: A classifier is a function $d(x)$ defined on $\mathcal{X}$ such that for every x, $d(x) \in \{1, 2, \dots, J\}$.
- Classification Trees are designed for **categorical response variables**.



2.2. Tree Classifiers

- Tree classifiers recursively split subsets of $\mathcal{X}$ into two descendant subsets (partition the space)
- Internal nodes represent decision splits (split the data along the way), terminal nodes contain final predicted class labels (e.g., 1, 2).
- Splits are formed based on conditions on $x_i \in x = (x_1, x_2, \dots)$.
- *Note: If you use a dummy 0/1 variable, you can still use it in the left-hand side (LHS) even if you arleady used it in the RHS. There's still variation!*



Terminal nodes:

- Objective: form **homogeneous** subgroups (pure classes) while maximizing external heterogeneity or obtain at least n objects.

2.3. Tree Structured Classifiers

- Constructing a tree requires:
  1. Dividing $\mathcal{X}$ into distinct, non-overlapping regions (select the splits and declare a termina lnode).
  2. Assigning each terminal node to a class.
- Select splits to increase "purity" of resulting subsets with respect ro the purity of the parent subset.

Steps:

- Define impurity metric (e.g., GINI)
- For every possible split: compute impurity reduction (objects whitin the subset are purer).
- Choose split that improves impurity most.
- Assign each terminal node t to a class j_0 such that:

$$p(j_0 \mid t) = \max_j p(j \mid t)$$

- *You assign a node to the most frequent class in that node: this reduces classification error.*

Additional notes:

- Split the full dataset. Imagine you have 4 independent dummy variables. First, split the data based on the values of the first variable x_1 (e.g., dummy: 0 or 1). Compute the degree of purity in each resulting subset (left and right). Check how much the overall purity increases due to the split.
  Try this for all 4 variables and select the one that leads to the highest increase in purity this variable will be used at the first node.
  Then, repeat the same procedure for the second level: in each of the two resulting subsets, consider the remaining variables for the next split.
  For example, imagine that in the left subset the best split is on x_2, and in the right subset it is on x_4. Then, in the left subtree you are left with x_3, x_4, and in the right subtree with x_2, x_3.
  Remark: As the sample size increases, you can explore all combinations of variable splits.
  Remark: The structure of the tree will reveal which variables are the most important for prediction.

- Same tree logic as classification, but terminal node value is the **mean** of THE OUTCOME (Y) observations.
- **Example:** Predict log-salary of baseball players based on years played and hits.

Explanation:

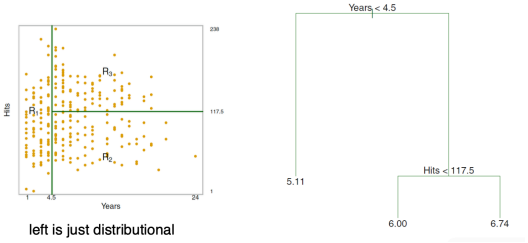
- Each leaf gives the average outcome of all observations in that region.
- Final group classification is an average: he leafs are created for group members to be similar, while maximizing the differences across groups.

- The tree shown has:
  - 2 internal nodes (splits),
  - 3 terminal nodes (leaves).
- Within each leaf: prediction = mean of observed outcome.
- Tree partitions data into:

$$R_1 = \{x \mid \text{Years} < 4.5\}$$
$$R_2 = \{x \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$$
$$R_3 = \{x \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$$

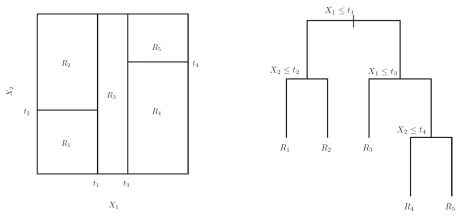
2.4. Regression Trees: Visualization

- Left plot: visual partition of $x = (\text{Years}, \text{Hits})$ space (distributional).



left is just distributional

Note: Even if observations are close in covariates, average outcomes across leaves can differ a lot.



2.5. Regression Trees: Tree Construction

- Goal: Estimate regression functions flexibly with fovus on strong out of smaple predictive power!
- Two main steps:
  1. Divide space $x = (x_1, \ldots, x_p)$ into J non-overlapping regions $R_1, \ldots, R_J$
  2. Predict average of Y in each R_j
- Theoretical research focuses on best way to construct the R_j. See below for more

- At each step, select a predictor x_j and a cutpoint s to split the space:
  $$\{x \mid x_j < s\} \quad \text{and} \quad \{x \mid x_j \geq s\}$$
  so as to minimize the Residual Sum of Squares (RSS).
- Consider **all** predictors $x_1, \ldots, x_p$ and **all** possible split values s for each.
- **Formally:** For each pair (j, s), define (one varaibles is used to create two splits):
  $$R_1(j,s) = \{x \mid x_j < s\}, \quad R_2(j,s) = \{x \mid x_j \geq s\}$$
  Then choose j and s that minimize:
  $$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$
- $\hat{y}_{R_k}$ is the mean of y_i in region R_k ($k = 1, 2$).
- *You can reuse the same variable later with different cutoffs, e.g., $x_1 < 5$, then $x_1 < 10$.*
- *The higher a variable appears in the tree and the more often it is used, the more important it is for prediction.*
- **Procedure:**
  - Start at the top
  - Recurse: split data within each resulting node using same logic.
  - Continue until stopping rule is met (e.g., region size ≤ 5 obs).
- Final prediction for any x is the mean outcome $\hat{y}_{R_j}$ in the region x falls into.

- **The goal** is to minimize forecast error **within each region** so that observations in the same leaf are similar (i.e., more homogeneous).
- Trees are built **top-down** using a greedy method called **recursive binary splitting**:
  - At each step, choose the split that gives the largest reduction in RSS.
  - We do not consider future splits local optimal choice only.
- **Splits are univariate:** only one covariate is used per split.

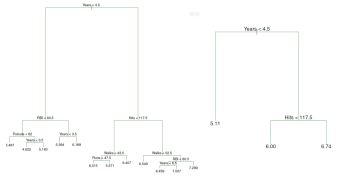
2.5.1. visualization

final nodes represent the partition representatiun un the x1 x2 space even though we have been recursive in the sense that at every stage of the construction of the tree we have concisdered only one variable in the end the whole tree structure allows for very complex and flexible interactions between variables. see left: the interaction terms are

super different depending on what is the joint distribution (the values of) x1 and x2. imagine in 3D is the outcome, how complex it can be! for every portion you can haev a differnt outcome! in the linear regresison case you would have a plane with a given angle (the coeff)

2.6. Overfitting

- The tree algorithm tends to overfit the data it splits the sample many times, fitting noise.
- Outcome: good predictions (low bias) on training data, but poor generalization (high variance) on test data.
  - This is what they call "overfitting".
- A smaller tree (fewer regions $R_1, \dots, R_J$) yields:
  - lower variance but
  - increased bias: classic bias-variance tradeoff.
- **Two solutions:**
  1. Build a smaller tree directly (more conservative split threshold, i.e., require large reduction in RSS). *Problem: may miss late good splits if early ones were bad.*
  2. Solution to the above: Build a large tree, then **prune** it by cutting back splits that dont reduce test error much. From the bottom put together splits that in the end are not so important to reduce the bias. This avoids you to lose important steps. You cut the internal node along all its internal branches. *This avoids losing important splits too early while only keeps those that really reduce errror in the data (and are not just due to idyosincrasy of a ery small subset of the data), but you may lose a lot in terms of bias*



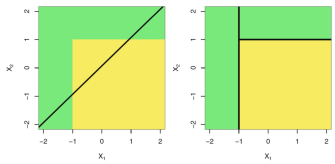
2.7. Pruning (reduce complexity of the model)

- **Goal:** Avoid overfitting by selecting a simpler subtree that balances fit and complexity.
- **G**row a large, fully expanded tree T_0 (possibly overfitting).
- **AIM:** Generate a subtree $T \subset T_0$ by pruning T_0 backward merging terminal nodes such that this subtree has the lowest test error rate.
- **HOW:** We select the optimal subtree $T \subset T_0$ by minimizing, given a fixed T, the penalized training error (as for lasso):

$$\sum_{m=1}^{|T|} \sum_{i\,:\,x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \qquad (2)$$

T is the number of terminal nodes. If that termianl node does not help, cut it! this is recursive, so you cut first layer of erminal nodes, then second layer ets

- **Interpretation of terms:**
  - $|T|$: number of terminal nodes (i.e., model complexity).
  - R_m: region (leaf) corresponding to terminal node m.
  - $\hat{y}_{R_m}$: mean response in region R_m.
  - α: tuning parameter penalizing complexity.
- **Tuning α:**
  - $\alpha = 0$: no complexity penalty select $T = T_0$
  - $\alpha \uparrow$: increasing penalty smaller tree
  - **Model selection:**
    - Use **cross-validation** to choose the value of α that yields the lowest **test error**.



2.8. Regression Trees vs. Linear Models

A **smooth** function vs. a **step** function:

$$\text{Regression model:} \quad f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

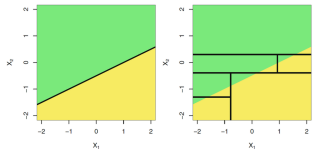
$$\text{Regression tree:} \quad f(X) = \sum_{m=1}^{M} c_m \cdot \mathbb{1}(X \in R_m)$$

Interpretation:

- The regression model fits a continuous linear surface.
- The tree partitions the space into subregions R_m, assigning the average outcome c_m in each.
- Each indicator $\mathbb{1}(X \in R_m)$ behaves like a dummy, assigning a given average to elements in that region.

Which model is better? It depends:

- If the true relationship is well approximated by a linear model, linear regression performs better.



- In highly non-linear, complex cases, trees may capture structure better.
- Estimating **test error** helps decide trees may overfit more easily due to high variance (and perform poorly in test set) ut sometimes there may be other considerations as well, such as interpretability or visualization.
- Tradeoff: regression = stable but possibly misspecified; tree = flexible but unstable.

Extra notes: If the relationship is linear, trees introduce unnecessary complexity. Trees are more sensitive to outliers, which can create overfitting.

Advantages and Disadvantages of Trees

Advantages

- For small trees: easy to explain, interpret and visualize.
- Some argue: decision trees mirror human decision-making more closely.

Disadvantages

- Hard to incorporate established theory into the algorithm.
- Interpretability shrinks with tree size (Its harder to isolate the effect of any single variable X_j on the output Y).
- Trees can be very non-robust (high variance). Small change in the data can cause a large change in the tree.

3. Bagging and Random Forests

Bagging (Bootstrap Aggregation)

- **Goal:** Improve predictive performance by aggregating many decision trees.
- **Idea:** Average over many fitted models to reduce variance and improve generalization.
- **Procedure:**
  - Draw B bootstrap samples (with replacement) from the training set.
  - Puild separate predictions using each sample
  - Predict by averaging the predictions across all B trees. Algorithms make a number of passes over the data. The ultimate results of interest are the collection of all the results from all passes.
- **The number of trees** B is not a critical tuning parameter: large B does not lead to overfitting (obv).
- In practice: choose B large enough for prediction error to stabilize.

Why Bagging Works (Berk, 2008)

- Averaging over fitted values reduces overfitting. The average cancels out the results shaped by idiosyncratic features 8outliers) of the data.
- Bias-variance trade-off is improved: combining trees reduces variance without increasing bias.
  - **Each tree:** high variance, low bias (Decision trees suer from high variance: if we split the training data randomly into two parts and fit a decision tree to each half, we could obtain quite dierent results)
  - **Bagged prediction:** low variance, low bias.
- Sharp decision boundaries from individual trees are smoothed (we take a bounddaty that is an average).

Limitations of Bagging

- If a strong predictor dominates the data, it will appear in the top split of many trees.
- All trees may look similar, leading to highly correlated predictions.
- Averaging highly correlated trees does not reduce variance effectively.

3.1. Random Forests

3.1.1. Trees Variability

- **Random Forests** (Breiman, 2001) provide a way to *de-correlate the trees*.
- **As in bagging:** we build a number of decision trees on bootstrapped training samples.
- **However,** when building the trees, at each split:
  - A **random sample of** m predictors is drawn from the full set of p predictors.
  - The split is only allowed to consider these m predictors.
  - A new random sample of predictors is drawn at each split.
- **Hence:** by building a random forest, the algorithm is not allowed to consider all available predictors at each split. This reduces correlation between trees.

noteice for m = p, random forest = bagging. Using a small value of m will typically be helpful when we have a large number of correlated predictors

Solution: Variable Importance via RSS Reduction

Goal: Assign a **score of predictability** to each predictor to quantify its contribution to prediction accuracy (also called "variable importance").

- Use the **RSS** (Residual Sum of Squares), as defined in Equation (1).
- **Method:** For each tree, at every node, record the total reduction in RSS that results from splits using a specific predictor. Then:
  - Aggregate these reductions over all B trees in the forest.
  - The result is the **total contribution of that predictor** to decreasing prediction error.
  - A **large value** indicates the predictor is important for reducing RSS and thus for accurate prediction.
  - The variable has played an important role in building the model (in reducing the RSS).
- Trick: Set all scores **relative to the largest** one:
  - Most important variable = 100.
  - Other variables scaled accordingly (e.g., 50 means "50% as important as th emost importatn one"). The idea is that it is difficult to interpret the reducion in teh RSS so you express the imrptant ce of all vars as a fracitonf of the imprtance of the avraibel with the highest score
  - Useful for interpretation and visualization.

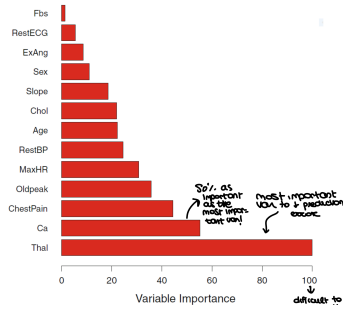


Figure 1. Enter Caption

4. HERE THE FOCUS CHANGES! Heterogeneous Treatment Effects

THE FOCUS NOW IS ON CAUSALITY

4.1. Motivation and Setup

- Until now, the goal was to predict the outcome Y.We have seen how regression trees produce a partition of the population according to covariates, whereby all units in a partition receive the same prediction.
- Now we shift to causal inference: estimating heterogeneosu **treatment effects** based on covariates. We want to study the dsitribtion of the TEs across subgroups that we haev not predefined. We want to let the algo find the subgroups that will maximize the variance of the ATE across subgroups.
- **Athey and Imbens (2016)** build on regression trees to estimate **heterogeneous treatment effects (HTEs)** using covariates.
  - A tree is built such that each leaf contains both treated and untreated observations
  - conditional on the lead the treatmetn is randoly assigned (needs unconfoudedness), we have put togetehr T and G with very similar covariaets as a result of the leaf geenration process (a sort of matching)
  - As they are in the same leaf, their characteristics are very similar (CIA) as if they were randomly assigned to treatment and control groups (based on unconfoundedness). This is the ATEx fo each leaf, we haev mno counterfactual and we are usign a matched contorl (what the treatemtn iunit would have had as outcome under t0).
  - HTEs are then estimated **within leaves** as the difference in average outcomes between treated and control.

- This gives a *matching estimator*: each leaf matches similar units with different treatments.
- They estimate heterogeneous treatment effects across leaves, while the treatment effects are uniform within leaves.

4.2. Causal Trees: Objectives

- Two key objectives:
  1. Estimate heterogeneity in causal effects (experimental or observational).
  2. Conduct valid inference about differences in treatment effects across subgroups (p-acking not solved by agnostic commitment to causal trees (better than pre-analysis plan)) + about inference effort to compute the SE. In comparison to prediction-based approaches, we are interested in preserving the validity of confidence intervals constructed on treatment effects within subgroups
- Especially useful when:
  - Many covariates, fewer observations; Imagine yuou need to find if a large numebr of covariaets as an eheteogenoeus treatemnt effect (or th einteracoitn of many covariates). Already with 10 dummies, you have 1024 possible combinaitons. Causal trees automatically identify covariates that explain treatment effect heterogeneity.No need to prespecify interactions or manually stratify.
  - Functional form of treatment effect unknown.

4.3. Causal Tree Estimator: Notation

- Π: a tree (that is, a partiotion of the feature space)
- $\#(\Pi)$: Number of leaves.
- $\ell(x, \Pi)$: Leaf containing x. This defiens the ampping leaf (numbered) - covariate.
- Hence we can write: $\Pi = \{\ell_1, \dots, \ell_{\#(\Pi)}\}$, with $\bigcup_{j=1}^{\#(\Pi)} \ell_j = \mathcal{X}$
- $D_i \in \{0, 1\}$: Treatment indicator.
- $\mathcal{S} = \mathcal{S}_{\text{treat}} \cup \mathcal{S}_{\text{control}}$: Training sample.
- Triplet for each observation: (Y_i^{obs}, X_i, D_i).

4.4. Estimation

- Given a tree P, define for all X and treatment levels D the population average outcome:

$$\mu(D, X; \Pi) = \mathbb{E}[Y_i \mid D_i = D, X_i \in \ell(X; \Pi)]$$

- Estimate (the average of the outcome for observations with the same T status and with covariates that will make them fall into the leaf selected by those values x of the covs):

$$\hat{\mu}(D, X; \mathcal{S}, \Pi) = \frac{1}{\#\{i \in \mathcal{S}_D : X_i \in \ell(X; \Pi)\}} \sum_{i \in \mathcal{S}_D : X_i \in \ell(X;\Pi)} Y_i^{obs}$$

- CATE:
  - Denote by $\tau(X; \Pi)$ the ATE conditional on a given tree, which is given by:

$$\tau(X; \Pi) \equiv \mathbb{E}[Y_{1i} - Y_{0i} \mid X_i \in \ell(X; \Pi)] = \mu_1(X; \Pi) - \mu_0(X; \Pi)$$

  - **With its estimated counterpart**:

$$\hat{\tau}(X; \Pi) \equiv \hat{\mu}(D = 1, X; \mathcal{S}, \Pi) - \hat{\mu}(D = 0, X; \mathcal{S}, \Pi)$$

  - *ATE estimate is the difference in the estimated average of the outcome of the treated and controls in the same leaf.*
  - *Note: Everything is conditional on* Π*, i.e., on the tree structure (the partitioning).*
  - *Note:* The idea of adding more homogeneous treatment effects (TEs) within each leaf is complex because we do **not observe individual treatment effects (ITEs)**.
  - **Key implication:** To approximate this goal, we try to **minimize the variance of the ATE estimate** = Obtain **precise ATE estimates** with low standard errors → the variance of the ATE depends on the outcome variance in the treated and control groups = A well-formed leaf will have low standard errors if the variance of outcomes is small for both treated and control observations in that leaf.
- **Interpretation:** Wanting homogeneous treatment effects within leaves serves a dual purpose:
  - From a **microeconometrics perspective**: increases the **statistical power** of the analysis by improving precision (power = probability of detecting a true effect). If the variance is low in T and C easier to detect treatment effect.
  - From a **machine learning perspective**: aligns with the goal of **minimizing prediction error** within each leaf.

4.5. Splitting Criterion: EMSE

- We need to give the tree a criterion for splitting → Partitioning criterion is set to maximize heterogeneity **across leaves** and minimize variance **within leaves** (want low bias).
- Objective

$$\widehat{\text{EMSE}}_\tau(\mathcal{S}^{\text{tr}}, N^{\text{est}}, \Pi) \equiv \underbrace{\frac{1}{N^{\text{tr}}} \sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi)}_{\text{rewards heterogeneity across leaves}} - \underbrace{\left(\frac{1}{N^{\text{tr}}} + \frac{1}{N^{\text{est}}}\right) \sum_{\ell \in \Pi} \left(\frac{S^2_{\mathcal{S}^{\text{tr}}_{\text{treat}}}}{p} \right.}_{\text{penalizes variance wit}}$$

Where:

- N^{tr} is the train sample
- $p = \frac{N_{\text{treat}}}{N}$ is the share of treated units.
- $S^2_{\mathcal{S}^{\text{tr}}_w}$ is the within-leaf variance of treatment effect estimates.
- $\sum_{i \in \mathcal{S}^{\text{tr}}} \hat{\tau}^2(X_i; \mathcal{S}^{\text{tr}}, \Pi)$: rewards heterogeneity across leaves.
- $S^2_{\mathcal{S}^{\text{tr}}_w}(\ell)$: penalizes variance within each leaf.

Interpretation:

- The blue term grows when the estimated ATEs differ strongly across leaves.
- The red term grows when there's high variance in treatment effects *within* leaves.
- Goal: maximize heterogeneity across leaves, minimize noise within each leaf (precision).

4.6. Causal Tree: Honesty

The point is that bagging uses the full data!

- So far we have split the data in training (N^{tr}) and test data (N^{test})
- Many existing machine learning methods cannot be used for constructing confidence intervals because methods are "adaptive" – they use the training data for model selection and estimation. The model dpedens on the data compositon you rsample, the jey advantage: use data as much as you can to learn. Issue: when talking about the statistical uncertainty of the estiamted parameter, you also ened to build in the fact that there is uncertianty related to the smapel that has affected the first step!
- Spurious correlation b/w covariates and outcomes affects the selected model. This leads to bias, which disappears only slowly with growing sample size
- Athey and Imbens (2016) propose an alternative approach they refer to as **honesty** within the training sample

Definition: Honesty

A model is "honest" if it does not use the same information for **selecting the model structure** (grow a tree) as for **estimation** given a model structure.

- ⇒ *instead of splitting my initial sample into training and test samples, I am willing to pay the cost of ¡ the training sample by introducing a sample used for estimation.*

- We further split the training sample into two parts:
  1. N^{tr} observations for model selection: used to construct the tree (splitting, cross-validation, etc.)
  2. N^{est} observations for estimating the treatment effects within each leaf

5. Causal Forest

- **The honest causal forest** is a random forest made up of honest causal trees
  - *Causal forests are **also a way to limit the risk of overfitting**, together with causal trees!*
- The random forest part is as before (bagging, picking a subset of predictors, averaging across many trees, etc.)
- The main contribution of Wager and Athey (2018) is an **asymptotic normality theory** for causal forest predictions which enables statistical inference
- We can then obtain confidence intervals, p-values, etc.

The method is analogous to random forests: Generate B causal trees and average their predictions such that

$$\hat{\tau}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{\tau}(x;\Pi_b)$$

- *The best predictor of the TE for an obs with values of covariates x will be an average over all the B trees of this prediction.*
- *⇒ obs with given values of X (x) will have B predictions in each tree ⇒ you average those out.*

Under some standard assumptions for consistency:

$$\frac{\hat{\tau}(x)-\tau(x)}{\sqrt{\text{Var}[\hat{\tau}(x)]}} \Rightarrow \mathcal{N}(0,1)$$

- *Asymptotic distribution of the ATE estimator in causal forests.*

and the asymptotic variance of causal forests can be accurately estimated.

Causal Forests: Details

Previously learned concepts (regression trees → random forests) all apply to causal trees → causal forests:

- A **causal tree can be visualized**, but a **causal forest cannot** — **variable importance graphs** help investigate the role of single predictors
  - *Important to do this to understand sources of TEH!*
- Similar to random forests, the advantage over a single tree is that it is **not always clear what the "best" causal tree is**
- By **averaging the treatment effects of many trees**, we **reduce variance and smooth sharp decision boundaries**

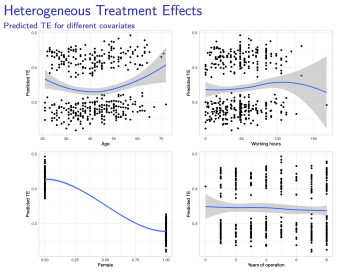


Figure 2. Enter Caption

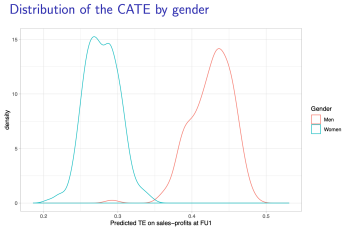


Figure 3. Enter Caption

Supplementary Concepts

Prediction vs Inference

- Machine learning and microeconometrics are different worlds.
- In statistical learning:
  - Independent variables X: called *input variables*, *features*, or *predictors*.
  - Dependent variable Y: often called *output* or *response*.
- Statistics → **Prediction**, Economics → **Inference**.
- Assume $Y = f(X)$:
  - **Prediction:** Predict $\hat{Y} = \hat{f}(X)$ where $\hat{f}$ is trained to make accurate predictions. The form of $\hat{f}$ is not important.
  - **Inference:** We care about understanding how Y changes with X, so we examine $\hat{f}$ itself. Prediction is a secondary concern.

Machine-Learning ML Terminology

- Models are *trained*, not estimated.
- Prediction problems:
  - **Supervised learning:** Observe both X and Y (e.g., regression/classification).
  - **Unsupervised learning:** Only X is observed, goal is to discover structure in X (e.g., clustering, NLP).
  - *Note: You discover the structure with an underlying Y.*

Some Notation: Data

- To avoid overfitting: fit on training set, test on independent data.
- Distinction:
  - **Training data** (N^{tr}): used to estimate $\hat{f}$.
  - **Test data** (N^{test}): used to evaluate $\hat{f}$.
- Compare $\hat{Y}^{tr}$ with Y^{test} using an error function.
  - **Overfitting:** Low error on training, high error on test.
  - *Intuition:* If a model works too hard to fit idiosyncratic patterns in training data that happen dby chance, those patterns likely wont generalize.

Appenidx

6. Bagging and Random Forests

Out-of-Bag

Out-of-Bag Error Estimation:

- With bagging and random forests, trees are repeatedly fit to (bootstrapped) subsets
- Assume that on average, each bagged tree makes use of around $2/3$ of the observations
- The remaining $1/3$ are out-of-bag (OOB) observations
- Predict the single response for the ith observation using each of the trees in which that observation was OOB by averaging the $B/3$ predictions for ith observation
- Compute OOB MSE over the single responses of all observations