

Generalized Random Forests

Introduction

Federico Nutarelli

Bocconi University

Table of Contents

Generalized Random Forests

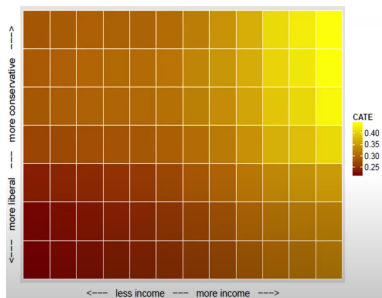
User-friendly set-up for R

Generalized Random Forests

User-friendly set-up for R

Why study GRF?

- Determine how much people like "Assistance to poor" politics, vs "Welfare" politics;
- "Assistance to poor" = "Welfare", but "Welfare" usually associated to laziness in USA



High heterogeneity: for more liberal and poorer the effect is 0.25; for more conservative and richer the effect is 0.4

It would be amazing if we could partition the covariate space $X_1 - X_2$ based on the heterogeneity of treatment effects (CATE) as displayed in Fig.1

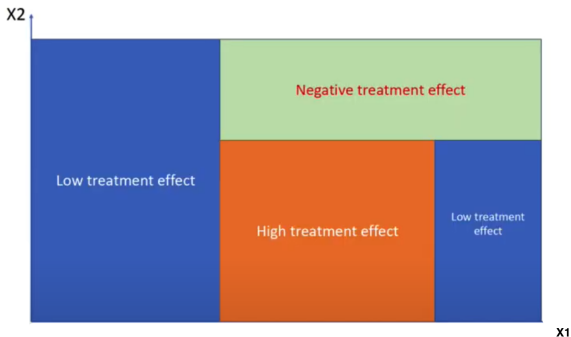


Figure: $X_1 - X_2$ ideally divided into regions according to $\tau(X)$

...however, we cannot observe the counterfactuals directly, which prevents us from computing the τ_i .

In reality, the **counterfactuals are missing**

	$Y^{W=1}$	$Y^{W=0}$	τ_i	W_i	X_i
1	?	$y_1^{W=0}$?	0	x_1
2	$y_2^{W=1}$?	?	1	x_2
3	?	$y_3^{W=0}$?	0	x_3
...
n	$y_n^{W=1}$?	?	1	x_n

Hence, we cannot employ a random forest directly to partition the covariate space based on τ_i (no training examples).

- the naive approach: train two random forests to create predictions for labels Y^0 and labels Y^1 , say $\hat{Y}^{RF}(W = 1, X = x)$ and $\hat{Y}^{RF}(W = 0, X = x)$ for a treatment assignment W
- Take the difference of the two predictions to create:

$$\hat{\tau}^{RF} = \underbrace{\hat{Y}^{RF}(W = 1, X = x) - \hat{Y}^{RF}(W = 0, X = x)}_{\text{WRONG}}$$

Why WRONG? Because we **used the same data** to decide **how to partition** into leaves and how **to estimate** mean values within each leaf
 → OVERFITTING.

Sounds familiar?

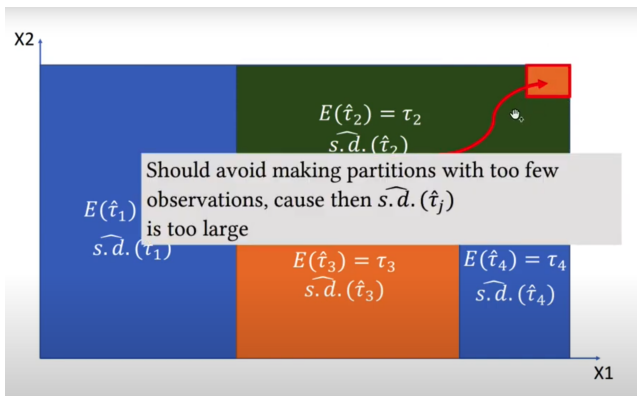
- the naive approach: train two random forests to create predictions for labels Y^0 and labels Y^1 , say $\hat{Y}^{RF}(W = 1, X = x)$ and $\hat{Y}^{RF}(W = 0, X = x)$ for a treatment assignment W
- Take the difference of the two predictions to create:

$$\hat{\tau}^{RF} = \underbrace{\hat{Y}^{RF}(W = 1, X = x) - \hat{Y}^{RF}(W = 0, X = x)}_{\text{WRONG}}$$

Why WRONG? Because we **used the same data** to decide **how to partition** into leaves and how **to estimate** mean values within each leaf
 → OVERFITTING.

Sounds familiar? **Honesty**

We want an unbiased estimator of τ_i (i.e. $\mathbb{E}[\hat{\tau}_i] = \tau_i$), with small standard errors to avoid tiny regions:



Generalized Random Forests (GRF) try to achieve these goals

Generalized Random Forests (GRF): aim

GRF are **causal forests**

General aim

Generalize classical random forests for estimating $\theta(x)$ by minimizing a moment condition as follows:

$$\mathbb{E}[\psi_{\theta(x), \nu(x)}(O_i) | X = x] = 0 \quad \text{for all } x \in X$$

where $\psi(\cdot)$ is a scoring function and $\nu(x)$ a nuisance parameter

Example of $\psi(\cdot)$? Maximum Likelihood Estimation:

$$\psi_{\theta(x), \nu(x)}(O) = \nabla \log(f_{\theta(x), \nu(x)}(O_i))$$

where $f_{\theta(x), \nu(x)}$ is the conditional distribution of O_i on X_i .

In other words, our objective is to identify a splitting criterion that **maximizes heterogeneity while minimizing the expected value of the scoring function**.

In order to achieve this, Athey et al. (2018) retain certain aspects of traditional random forests, such as recursive partitioning, sub-sampling, and random split selection.

However, they depart from the notion that the final estimate is an average of the leaf members. More specifically, they consider forests as a form of adaptive nearest neighbors by intelligently defining weights (outlined below).

How are splits chosen? (Practice)

The process begins by drawing a random subsample from the complete dataset, without replacement.

This subsample is used to create a single root node. The root node is then divided into child nodes, and this division is repeated recursively to construct a tree.

The procedure terminates when no further nodes can be split.

The main difference between GRF's approach to growing trees and that of classic random forests is in **how the quality of a split is measured**.

With causal effect estimation, the goodness of a split relates to **how different the treatment effect estimates are in each node**

Theoretical foundations for such splitting strategy in next slides...

How are splits chosen? (Theory)

Set up

- $O_i = \{Y_i, W_i\}$ where Y_i is the outcome and W_i the exogenous treatment
- Define $\alpha_i(x)$ as being similarity weights measuring the relevance of the i^{th} training example to fitting $\theta(\cdot)$ at x

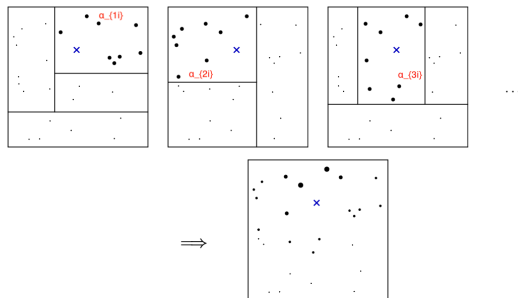
Then:

$$\left(\hat{\theta}(x), \hat{\nu}(x)\right) \in \operatorname{argmin}_{\theta, \nu} \left\{ \left\| \sum_{i=1}^n \alpha_i(x) \psi_{\theta, \nu}(O_i) \right\|_2 \right\}.$$

when the above expression has a unique root, we can say that $(\hat{\theta}(x), \hat{\nu}(x))$ solves $\sum_{i=1}^n \alpha_i(x) \psi_{\hat{\theta}(x), \hat{\nu}(x)}(O_i) = 0$ being $\psi(\cdot)$ some scoring function.

How are splits chosen? (Theory)

How are weights α_i determined?



Each tree starts by giving equal (positive) weight to the training examples in the same leaf as our test point x of interest, and zero weight to all the other training examples. Then, the forest averages all these tree-based weightings, and effectively measures how often each training example falls into the same leaf as x .

How are splits chosen? (Theory)

Formally:

$$\alpha_{bi}(x) = \frac{\mathbf{1}(\{X_i \in L_b(x)\})}{|L_b(x)|}, \quad \alpha_i(x) = \frac{1}{B} \sum_{b=1}^B \alpha_{bi}(x).$$

For regression trees:

- Aim: estimate $\theta(x) = \mu(x) = \mathbb{E}[Y_i | X_i = x]$ (Notice: reg. trees estimate the labels Y_i and not the causal effect $\theta(x)$)
- $\psi_{\mu(x)}(Y_i) = Y_i - \mu(x)$
- $\hat{\mu}(x) \in \arg \min_{\mu} (\|\sum_{i=1}^n \alpha_i(Y_i - \mu(x))\|_2)$
- take empirical version of $\|\cdot\|_2$ and minimize it as $\sum_{i=1}^n = \frac{1}{B} \sum_{b=1}^B \alpha_{bi}(x)(Y_i - \hat{\mu}(x)) = 0$, which implies $\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^B \hat{\mu}_b(x)$ where $\hat{\mu}_b(x)$ is the prediction made by a single CART regression tree.

How are splits chosen? (Theory, pt.2)

Aim: Finding trees that when combined into a forest induce weights $\alpha_i(x)$ leading to good estimates of $\theta(x)$.

Splitting rule: splitting to maximize heterogeneity

Just like in Breiman's forests, our search for good splits proceeds greedily, i.e., we seek splits that immediately improve the quality of the tree fit as much as possible. Every split starts with a parent node $P \subseteq \mathcal{X}$; given a sample of data \mathcal{J} , we define $(\hat{\theta}_P, \hat{\nu}_P)(\mathcal{J})$ as the solution to the estimating equation, as follows (we suppress dependence on \mathcal{J} when unambiguous):

$$(4) \quad (\hat{\theta}_P, \hat{\nu}_P)(\mathcal{J}) \in \operatorname{argmin}_{\theta, \nu} \left\{ \left\| \sum_{\{i \in \mathcal{J}: X_i \in P\}} \psi_{\theta, \nu}(O_i) \right\|_2 \right\}.$$

We would like to divide P into two children $C_1, C_2 \subseteq \mathcal{X}$ using an axis-aligned cut such as to improve the accuracy of our θ -estimates as much as possible; formally, we seek to minimize $\operatorname{err}(C_1, C_2)$ defined as $\operatorname{err}(C_1, C_2) = \sum_{j=1,2} \mathbb{P}[X \in C_j | X \in P] \mathbb{E}[(\hat{\theta}_{C_j}(\mathcal{J}) - \theta(X))^2 | X \in C_j]$, where $\hat{\theta}_{C_j}(\mathcal{J})$ are fit over children C_j in analogy to (4), and expectations are taken over both the randomness in $\hat{\theta}_{C_j}(\mathcal{J})$ and a new test point X .

In other words...

- we start with an estimate of $\theta(x)$ given by $(\hat{\theta}_P, \hat{\nu}_P)$;
- we then try to improve the latter more and more at each split. How?
- minimizing $err(C_1, C_2)$, which accounts not only for the probability that $X \in C_j$ but also for the error $\hat{\theta}_{C_j} - \theta(X)$

Would be great to find an axis align split minimizing $err(C_1, C_2)$ at every split!

PROBLEM: we do not observe $\theta(X)$! \rightarrow another big difference w.r.t. regression trees (where the aim was prediction and the error is defined on the observed Y_i).

We need to re-elaborate the target criterion

PROPOSITION 1. *Suppose that basic assumptions detailed in Section 3 hold, and that the parent node P has a radius smaller than r for some value $r > 0$. We write $n_P = |\{i \in \mathcal{J} : X_i \in P\}|$ for the number of observations in the parent and n_{C_j} for the number of observations in each child, and define*

$$(5) \quad \Delta(C_1, C_2) := n_{C_1} n_{C_2} / n_P^2 \left(\hat{\theta}_{C_1}(\mathcal{J}) - \hat{\theta}_{C_2}(\mathcal{J}) \right)^2,$$

where $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$ are solutions to the estimating equation computed in the children, following (4). Then, treating the child nodes C_1 and C_2 as well as the corresponding counts n_{C_1} and n_{C_2} as fixed, and assuming that $n_{C_1}, n_{C_2} \gg r^{-2}$, we have $\text{err}(C_1, C_2) = K(P) - \mathbb{E}[\Delta(C_1, C_2)] + o(r^2)$ where $K(P)$ is a deterministic term that measures the purity of the parent node that does not depend on how the parent is split, and the o -term incorporates terms that depend on sampling variance.

Hence, we can consider splits that make the above Δ -criterion large! (enters with minus sign).

Gradient tree algorithm

PROBLEM \rightarrow optimizing $\Delta(C_1, C_2)$ while finding $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$ is computationally expensive).

SOLUTION \rightarrow optimizing an alternative criterion $\tilde{\Delta}(C_1, C_2)$ build using gradient approximations of $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$,

$$\tilde{\theta}_C = \hat{\theta}_P - \frac{1}{|\{i : X_i \in C\}|} \sum_{\{i: X_i \in C\}} \xi^T A_P^{-1} \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i)$$

- $A_P \rightarrow$ gradient of $\psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i)$;
- $\xi^T A_P^{-1} \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i) \rightarrow$ influence function of the i^{th} observation for computing θ_P in the parent

What is $\tilde{\theta}_C$? 2 semi-formal ways

Two ways-
to explain

Remember that $\hat{\theta}_C = \arg \min_{\theta} \left\{ \left\| \sum_{\{i \in \mathbb{J}: X_i \in C\}} \psi_{\theta, \nu}(O_i) \right\|_2 \right\}$. Being an $\arg \min_{\theta}$ it can be approximated using sub-gradient methods. Specifically: $\theta^{t+1} = \theta^t - \gamma \nabla \psi_{\hat{\theta}^t, \hat{\nu}^t}$. In our scenario, θ^{t+1} is the θ of the next node, i.e. the θ of the child, while θ^t is the θ of the previous node (the parent). So: $\theta^C = \theta^P - \gamma \nabla \psi_{\hat{\theta}^P, \hat{\nu}^P}$ where $\gamma = \frac{1}{\sum_{\{i: X_i \in C\}} \xi^T}$

It can be shown that for a general estimator $\phi(P)$, an approximation using influence function can be found as $\phi(P) \approx \phi(P^0) + \mathbb{E}[IF(X)]$ being $IF(X)$ an influence function. In our set-up, $\tilde{\theta}_C = \phi(P)$ and $\hat{\theta}_P = \phi(P^0)$

From now on we can assume (following Athey, 2018) that the latter is simply a gradient approximation of $\hat{\theta}_C$.

What we should really care about, however, is **why do we need a gradient approximation of $\hat{\theta}$?**

- if we wanted to optimize $\Delta(C_1, C_2)$ while finding $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$ we should have computed not only $\Delta(\cdot)$ for each possible split but also $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$ with (4) since they also appear as part of $\Delta(\cdot)$ (see highlighted part in slide 37)
- to avoid computing $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$ with (4) for every possible split we can re-write $\hat{\theta}_C$ with its gradient approximation and apply a recursive algorithm that avoids the calculation of $\hat{\theta}_{C_1}, \hat{\theta}_{C_2}$ with (4) for every possible split (see next slides to see how *)

How sub-gradient definition of $\hat{\theta}$ is useful: pt.1


Algorithmically, the above splitting scheme can be divided in 2 main Steps :

- **Labeling step:** Compute $\hat{\theta}_P, \hat{\nu}_P, A_P^{-1}$ and build pseudo-outcomes (sort of new y_i):

$$\rho_i = -\xi^T A_P^{-1} \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i) \in \mathbb{R}$$

- **Regression step:** run a CART regression split on ρ_i , i.e. split P in C_1 and C_2 to maximize

$$\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^2 \frac{1}{|\{i : X_i \in C_j\}|} \left(\sum_{\{i : X_i \in C_j\}} \rho_i \right)^2$$

Once we have executed the regression step, we relabel observations in each child by solving the estimating equation, and continue on recursively.  SCHOOL FOR ADVANCED STUDIES LUCCA

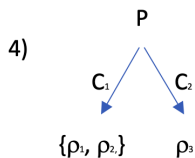
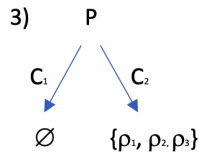
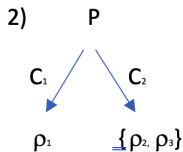
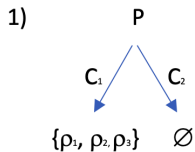
How sub-gradient definition of $\hat{\theta}$ is useful: pt.2

The idea is to re-formulate the problem of estimating θ_C in a prediction problem. This is done by labeling the observations with ρ_i which contains $\hat{\theta}$, and consider the problem as a regression problem solvable with CART.

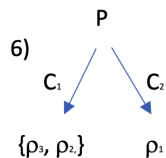
Example: say we have 3 observations $\{1, 2, 3\}$ with associated ρ_1, ρ_2, ρ_3 . For the sake of simplicity assume that $\rho \in [0, 1]$ (not always the case). In particular say: $\rho_1 = 0, \rho_2 = 0, \rho_3 = 1$. We first label observations according to ρ .

We want to maximize heterogeneity and to do so we should maximize consider all possible partitions of $\{1, 2, 3\}$ observations in C_1 and C_2 by maximizing $\tilde{\Delta}(C_1, C_2)$ (see next slide)

How sub-gradient definition of $\hat{\theta}$ is useful: pt.3



...



How sub-gradient definition of $\hat{\theta}$ is useful: pt.3

Compute $\tilde{\Delta}(C_1, C_2)$ for any possible permutation, e.g.: 1) $\tilde{\Delta}(C_1, C_2) = \frac{1}{2}$,
... 4) $\tilde{\Delta}(C_1, C_2) = 1$, ..., 6) $\tilde{\Delta}(C_1, C_2) = \frac{1}{2}$.

Case **4)** maximizes \rightarrow we pick up the split provided by 4).

We then proceed recursively using C as new parent node:

- Compute $\hat{\theta}_C, \hat{\nu}_C$ using (4) (where $P = C$ now); \leftarrow **KEY: we compute it once!***21
- Attribute pseudo-outcomes ρ_i (constructed using $\hat{\theta}_C, \hat{\nu}_C$) to observations
- apply regression step

Hence we compute $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$ **only once** (not for every permutation).

Notice: The key to do that was defining $\tilde{\theta}_C$ which allowed to define $\tilde{\Delta}(\cdot)$ and to apply a less computationally expensive algorithm.

Table of Contents

Generalized Random Forests

User-friendly set-up for R

- What are R-learners? R-learners are part of "meta-learners" which are algorithms leveraging on machine-learning predictive power to estimate a treatment effect $\tau(\cdot)$;
- In the case of R-learners, let $e(x) = P[W = 1|X = x]$ and $m(x) = \mathbb{E}[Y|X = x]$, the idea is to estimate them using ML and adopt their estimate to find $\hat{\tau}(\cdot)$
- GRF can be seen as an implementation of R-learners.

Recap on R-learners

- Assuming unconfoundedness, overlap ($\eta < e(x) \leq 1 - \eta$), and SUTVA (treating one unit does not affect others), Robinson (1988) found the following decomposition (regression):

$$Y_i - m(X_i) = (W_i - e(X_i))\tau^*(X_i) + \epsilon_i$$

- Goal of the R-learner? Find $\tau(\hat{\cdot})$ minimizing:

$$L(\tau) = \mathbb{E}[\{(Y_i - m(X_i)) - \tau(X_i)(W_i - e(X_i))\}^2]$$

Optimal $\tau(\cdot)$ is $\tau^*(\cdot)$

Why minimize $L(\tau)$ and not something else? (see next slide)

It can be shown that

$$L(\tau) - L(\tau^*) = \mathbb{E}[(\tau(X_i) - \tau^*(X_i))^2 \cdot (W_i - e(X_i))^2]$$

hence **by overlap** (start from overlap and perform smart multiplications):

$$\eta^2 \mathbb{E}[(\tau(X_i) - \tau^*(X_i))^2] \leq R(\tau) \leq (1 - \eta)^2 \mathbb{E}[(\tau(X_i) - \tau^*(X_i))^2]$$

i.e. we showed that the R-learner must minimize the difference between $\tau(X)$ and $\tau^*(X)$.

R-learner finds $\hat{\tau}(\cdot)$ based on 2 steps:

1. Use any ML method to estimate $\hat{e}(X)$ and $\hat{m}(X)$
2. Minimize $\hat{L}(\tau)$ to find $\hat{\tau}(\cdot)$, i.e. the $\hat{\tau}(\cdot)$ minimizing

$$\hat{L}(\tau) = \sum_{i=1}^n ((Y_i - \hat{m}(X_i)) \cdot \tau(X_i)(W_i - \hat{e}(X_i)))^2 + \Lambda_n$$

where Λ_n is a general regularizer.

Causal forests as implemented in `grf` is motivated by the R-learner.

- i. Concretely `grf` package starts by creating estimates $\hat{e}(\cdot)$ and $\hat{m}(\cdot)$ via **regression** forests.
- ii. It makes out-of-bag predictions (predictions are average outputs from trees whose training data, by honesty, did not include the i^{th} observation, i.e. a $(-i)$), i.e. we solve

$$\mathbb{E}[\alpha_i \psi_{\theta(x), \nu(x)}(O_i) | X = x] = 0 \quad \text{for all } x \in \mathcal{X}$$

where $\psi(\cdot)$ is taken from R-learner. The **key difference from R-learners is the forest weighting** α_i , i.e.:

$$\hat{\tau}(x) = \frac{\sum_{i=1}^n \alpha_i(x) (Y_i - \hat{m}^{(-i)}(X_i)) (W_i - \hat{e}^{(-i)}(X_i))}{\sum_{i=1}^n \alpha_i(x) (W_i - \hat{e}^{(-i)}(X_i))^2}$$

Quick comments:

- In the formula above, the x in $\tau(x)$ is the target test point;
- α_i , therefore is computed as we have seen:

$$\alpha_{bi}(x) = \frac{\mathbf{1}(\{X_i \in L_b(x)\})}{|L_b(x)|}, \quad \alpha_i(x) = \frac{1}{B} \sum_{b=1}^B \alpha_{bi}(x).$$

on the test point x